Cours PHP Accéléré

Version 1.0.5

Gérard Rozsavolgyi Sylvain Austruy

juin 10, 2020

Table des matières

CHAPITRE 1

Ce cours au format epub

Tuto PHP en accéléré format epub

CHAPITRE 2

Ce cours au format pdf

Tuto PHP en accéléré en pdf

CHAPITRE 3

Tables des matières :

3.1 Caractéristiques Principales

3.1.1 Langage interprété

— Pas de compilation
— Exécuté instruction par instruction
— Multi-plateformes
— Compilateur AOT/ByteCode en PHP7 Zend
— Compilateur JIT pour HHVM Facebook
— Langage Hack proposé par Facebook

3.1.2 Spécialisé dans la génération de texte ou de documents

— HTML
— PDF
— Images

3.1.3 Fichiers d'extension .php

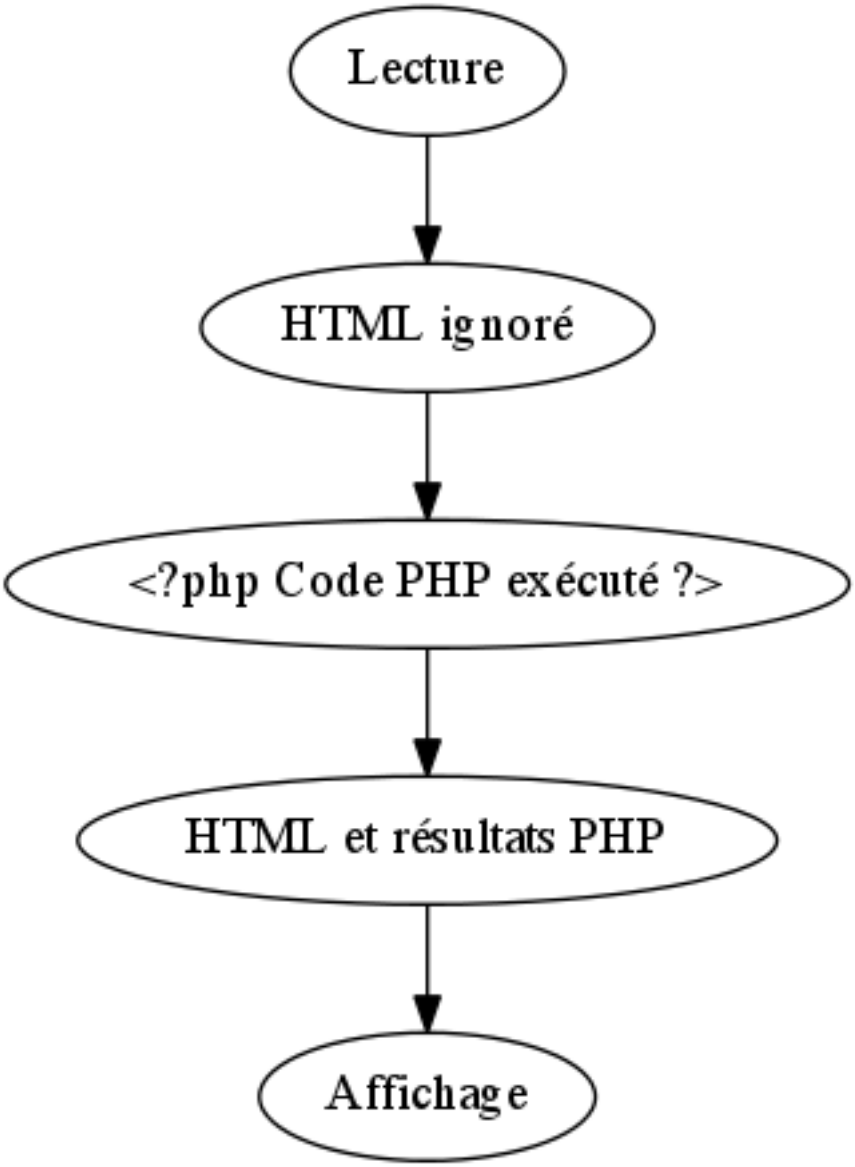
— Code inséré dans une page HTML
— Entre balises <?php et ?>

3.2 Fonctionnement

3.2.1 L'interpréteur

lit un fichier source .php puis génère un flux de sortie avec les règles suivantes :

- toute ligne située à l'extérieur d'un bloc PHP (entre <?php et ?>) est recopiée inchangée dans le flux de sortie
- le code PHP est interprété et génère éventuellement des résultats intégrés eux aussi au flux de sortie
- les erreurs éventuelles donnent lieu à des messages d'erreurs qu'on retrouve également dans le flux de sortie (selon la configuration du serveur)
- une page html pure sauvegardée avec l'extension .php sera donc non modifiée et renvoyée telle quelle ...



3.3 Historique

3.3.1 Créateur

Rasmus Lerdorf, un programmeur Groenlandais avec la nationalité canadienne, crée PHP en 1994 pour analyser les connexions sur son site Web. Il réalise les 2 premières moutures du langage (v1 et v2). En 1997, deux étudiants, Andi Gutmans et Zeev Suraski, reprennent le moteur, il en sortira PHP 3.0 puis les outils Zend.

Note : Le langage PHP a subi de profonds remaniements et a bénéficié de beaucoup d'enrichissements depuis ces premières versions. La première version objet de PHP (la version 4) a été profondément remaniée lors du passage de PHP4.0 à PHP5.0 et s'inspire désormais largement du modèle de Java.

La version actuelle de PHP est la 7.2, sortie en 2018. La version 7 est sortie en Décembre 2015 et il n'y a jamais eu de version 6 ! PHP 7 (https ://wiki.php.net/rfc/php7timeline)

Avertissement : Le début du développement de la version 6 de PHP date de 2005 et a été abandonnée en raison de difficultés d'intégration du support Unicode. Les autres fonctionnalités envisagées ont depuis été intégrées à PHP 5.3 ou 5.4. Ca n'a pas empêché un certain nombre de livres intitulés PHP 6 de sortir bien que cette version n'existera jamais . . .

3.3.2 La saga PHP 7 :

— Facebook a publié en 2011 HipHop Virtual Machine dit *HHVM*, une machine virtuelle permettant de pré-compiler le code PHP en bytecode à la manière de Java (JIT Compiler).
— En 2014, Facebook sort le langage Hack, s'exécutant grace à HHVM et apportant des fonctionnalités comme le typage fort, des classes paramétrables, une API de collections plus riche et cohérente, des traitements asynchrones et davantage de sécurité avec l'extension XHP.
— Le framework PHP Phalcon (extension PECL de PHP), propose Zephir (Zend Engine PHp Intermediate) (https ://github.com/phalcon/zephir) qui permet la création d'extensions rapides en PHP (codées en C) de manière simplifiée (sans écrire du C directement).
— La société Zend a réagi face à HHVM en accélérant le développement de la nouvelle mouture de PHP 7 et en améliorant ses performances avec un mécanisme de compilation AOT (Ahead of Time)

3.3.3 PHP

Signifie d'abord *Personal Home Pages* puis *HypertextPreProcessor*

3.3.4 Syntaxe et structure

— Proche du C ou du Perl
— Peu structuré au début
— Possibilité de définir des fonctions et des classes
— Très bonnes performances pour PHP 7 (améliorations de l'ordre de 50% de la vitesse d'exécution)

3.3.5 Utilisation

— Grand succès
— Utilisation par de très grands sites
— beaucoup de code libre disponible.
— des dizaines de millions de sites Web l'utilisent à travers le monde...
— Comme Le Monde (http ://lemonde.fr/) , Facebook (http ://facebook.fr/) ou Yahoo (http ://yahoo.fr/)

3.3.6 CMS

Les grands CMS Content Management Systems ou Systèmes de Gestion de Contenus utilisent PHP, comme :
— Wordpress
— Joomla
— Drupal

Les CMS de ecommerce aussi :

— Prestashop
— Magento

3.3.7 Frameworks

De grands Frameworks de développement Web aussi sont disponibles en PHP :
— Symfony
— Zend
— Laravel
— Phalcon
— CakePHP
— Yii
— Slim

Note : Plus de 300 millions de sites sont réalisés en PHP à travers le monde !

3.4 Installation PHP

Indication : Tout informaticien doit savoir ce que LAMP veut dire . . .

3.4.1 LAMP :

— Linux
— Apache
— MySQL
— PHP

Par exension, le logiciel équivalent pour Windows, s'est retrouvé nommé :

3.4.2 WAMP :

Pour Windows

Et MAMP pour les Macs. . .

3.4.3 MAMP :

Pour Mac. Pas indispensable car Apache et PHP sont installés sous Mac OS mais un peu plus simple à configurer. Il vous faudra alors installer MySQL ou MariaDB soit dans un paquet soit en utilisant *homebrew* ou *macports*

Pour toutes les plate-formes, on peut aussi installer XAMPP (https ://www.apachefriends.org/fr/index.html)

Revenons à l'OS de référence à présent.

3.4.4 Sous Linux :

— Installer Apache2
— Installer PHP7
— Télécharger éventuellement la documentation (paquet php-doc)
— Vérifier le fichier php.ini
— Installer MySQL (client et serveur)
— Installer PHPMyAdmin
— Installer des paquets complémentaires (dont les noms commencent par php7x-)

3.5 Exemples simples

3.5.1 bonjour

```
<!doctype html>
<html>
<head>
<meta charset="utf-8" />
<title> Bonjour depuis PHP </title>
</head>
<body>
<?php echo 'Bonjour généré dynamiquement en PHP !'; ?>
</body>
</html>
```

3.5.2 Résultat brut html

```
<!doctype html>
<html>
<head>
<meta charset="utf-8" />
<title> Bonjour depuis PHP </title>
</head>
<body>
Bonjour généré dynamiquement en PHP !</body>
</html>
```

3.5.3 Exécution

bonjour

3.5.4 Infos PHP

```
<?php phpinfo(); ?>
```

3.5.5 Exécution

infos php

3.5.6 User Agent

```
<!doctype html>
<html>
<head>
<title>
Navigateur
</title>
<body>
Les informations sur le Navigateur sont :
<?php
 echo $_SERVER['HTTP_USER_AGENT'];
 ?>
</body>
</html>
```

3.5.7 Exemple de Résultat

```
Les informations sur le Navigateur sont : Mozilla/5.0
(Macintosh; U; Intel Mac OS X 10_6_4; fr-fr) AppleWebKit/
↪533.18.1
(KHTML, like Gecko) Version/5.0.2 Safari/533.18.5
```

3.5.8 Exécution

User-Agent

Vous pouvez consulter la documentation de PHP sur :

doc php (http ://php.net/manual/fr/)

3.6 Imbrication de code HTML et PHP

On peut par exemple :

— Commencer une page HTML
— puis écrire un peu de PHP
— reprendre en HTML
— etc.

3.6.1 Exemple :

```
<!doctype html>
<html>
<head>
<title>
Navigateur
</title>
<body>
Les informations sur le Navigateur sont :
<?php
 $AGENT=$_SERVER['HTTP_USER_AGENT'];
 echo $AGENT;
 echo("\n<P>");
 if (stristr($AGENT,"MSIE")) {
          ?>
          <b>Vous semblez utiliser Internet Explorer !</b>
 <?php }
     elseif (preg_match("/Firefox/i",$AGENT))
 { ?>
         <b>Vous semblez utiliser Firefox !</b>
 <?php }
                  elseif (preg_match("/chrome/i",$AGENT))
 { ?>
         <b>Vous semblez utiliser Chrome !</b>
 <?php }
 elseif (preg_match("/Safari/",$AGENT))
          { ?>
         <b>Vous semblez utiliser Safari !</b>
  <?php }
  else echo "Navigateur Inconnu !";  ?>
</body>
</html>
```

ATTENTION : ça peut vite devenir **ILLISIBLE**

3.6.2 Exécution

user-agent php

3.6.3 Remèdes :

Pour ne pas écrire ce genre de code, quelques solutions courantes :

- Utiliser des fonctions PHP
- Utiliser des Classes et Objets PHP
- Séparer les modèles des Vues
- Séparer les Modèles, les Vues et les Contrôleurs (Modèle MVC)
- Utiliser des systèmes de templates comme Twig

3.7 Un formulaire HTML et sa réponse en PHP

On code ici :

— un petit formulaire HTML
— et sa réponse en PHP

3.7.1 Formulaire HTML

```
<!doctype html>
<html>
<head>
<meta charset="utf-8" />
<title>
Formulaire html
</title>
</head>
<body>
<form action="reponse.php" method="GET">
Votre nom :<input type="text" name="nom">
Votre âge :<input type="text" name="age">
<p>
<input type=submit value="Envoyer">
</form>
</body>
</html>
```

3.7.2 Sa réponse

```
<!doctype html>
<html>
<head>
<meta charset="utf-8" />
<title>
Test Formulaire PHP
</title>
</head>
<body>
<h1>Bonjour,  <?php echo $_GET['nom'] ?></h1>
<h2>Vous semblez avoir  <?php echo $_GET['age'] ?></h2>
   <?php
    $n = $_GET['nom'];
    $a = $_GET['age'];
    ?>
Votre nom est stocké dans la variable $n
dont le type est <?php echo gettype($n) ?>
```

(suite sur la page suivante)

(suite de la page précédente)

```
Votre âge est stocké dans la variable <b>$a</b>
<br/> dont le type est <i><?php echo gettype($a); ?></i>
<br/> On peut la transformer en <i>integer</i> en faisant :
   <?php settype($a,"integer"); ?>
<br/>
    Type de $a :<?php echo gettype($a); ?>
</body>

</html>
```

3.7.3 Exécution

Formulaire

3.8 Les variables en PHP

3.8.1 Déclaration simple :

```
<?php
$variable = "une variable en PHP";
// Une autre variable :
$Variable = 1000;
```

3.8.2 Existence de variables, la fonction isset() :

```
<?php
$a = "une variable en PHP";
if(isset($a)) echo "la variable a existe";
unset($a);
echo "la variable a a été supprimée ...";
```

3.8.3 Test de variables, la fonction empty() :

```
<?php
$a = "une variable en PHP";
if (!empty($a)) echo " La variable existe et elle n\'est␣
 ↪pas vide !";
```

Avertissement : La fonction empty() répond vrai si la variable n'existe pas et ceci sans faire aucun warning ! En outre, avant PHP 5.5, on ne peut pas l'utiliser sur autre chose que des variables (impossible d'appeller une fonction dans l'argument qu'on lui passe)

3.8.4 Test de variables en PHP 7 avec l'opérateur *coalescent*

L'opérateur *Null coalescent* **??** permet de simplifier certains tests d'existence de variables et d'alternatives, comme par exemple :

```
<?php

// $a non initialisée
$b = 143;

echo $a ?? 3; // affiche 3
echo PHP_EOL;
echo $a ?? $b ?? 7; // affiche 143
echo PHP_EOL;
```

Ce qui permet de limiter le recours à *isset* dans de nombreuses situations comme :

```
<?php
// Récupère la valeur de $_GET['email'] et  retourne 'nobody
↪'
// si elle n'existe pas.
$mail = $_GET['email'] ?? 'nobody@null';
// Equivalent à:
$mail = isset($_GET['email']) ? $_GET['email'] :
↪'nobody@null';

// Coalescing ?? peut être chainé :
// On renvoie la première valeur définie parmi
// $_GET['email'], $_POST['email'], et 'nobody@null.com'.
$mail = $_GET['email'] ?? $_POST['email'] ?? 'nobody@null';
echo "$mail\n";
```

3.8.5 Portée des variables :

— Par défaut, toutes les variables sont **locales**
— Leur portée se réduit à la fonction ou au bloc de leur déclaration
— Pour déclarer une variable globale, on peut utiliser le tableau $_GLOBALS[ ]

```
<?php $_GLOBALS['MaVar']="Bonjour"; ?>
```

3.8.6 Constantes :

```
<?php
define("USER","TOTO");
echo USER; // Notez l'absence de $ ici
```

3.9 Les chaînes en PHP

3.9.1 Les bases :

Guillemets ou Cotes :

```
<?php
$var="Hello PHP";
$machaine="le contenu de \$var est $var<br>";
echo $machaine;
//ou avec des '  ':
$mystring='le contenu de $var est '.$var;
echo $mystring;
```

dont le résultat sera toujours :

```
le contenu de $var est Hello PHP
```

La concaténation :

A l'aide de **.**

La longueur d'une chaine :

```
<?php int lg=strlen($chaine); ?>
```

Accéder au caractère i d'une chaine :

```
<?php echo $chaine[i]; ?>

La chaine est traitée comme un tableau indéxé par un *entier*
La plupart des tableaux de PHP sont indéxés par des chaines...
```

Mettre en majuscules/minuscules :

— avec *strtoupper()* pour obtenir des majuscules
— avec *strtolower()* pour mettre en minuscules
— avec *ucfirst()* pour mettre en majuscule la première lettre d'une chaine
— avec *ucwords()* pour mettre en majuscule la première lettre de chaque mot dans une chaine

3.9.2 Recherche de sous-chaines ou de motifs dans une chaine :

— avec *strstr()*
— avec *stristr()*
— avec *ereg()* ou *eregi()*

Par exemple :

```
<?php
$AGENT=$_SERVER['HTTP_USER_AGENT'];
echo $AGENT;
echo("\n<P>");
if (stristr($AGENT,"MSIE"))
   echo "Vous semblez utiliser Internet Explorer !</b>";
   elseif (ereg("Firefox",$AGENT))
          echo "Vous semblez utiliser Firefox !</b>";
   elseif (eregi("chrome",$AGENT))
          echo "Vous semblez utiliser Chrome !</b>";
```

Indication : Les variantes de ces fonctions comportant un i indiquent une insensibilité à la casse c'est à dire que les majuscules et minuscules sont considérées comme identiques.

Exemple : Test un peu plus complet du UserAgent :

```
<?php
function getBrowser($userAgent){
  if (preg_match("/MSIE(.{5})/i",$userAgent,$num))
    return $num[0];
  elseif (preg_match("/Firefox(.*)/i",$userAgent,$num))
    return $num[0];
  elseif (preg_match("/chrome(.{4})/i",$userAgent,$num))
    return $num[0];
  elseif (preg_match("/safari/i",$userAgent,$num)){
    preg_match("/Version(.{4})/",$userAgent,$num);
    return "Safari ".$num[0];
}
  else return "Navigateur Inconnu";
}
```

(suite sur la page suivante)

(suite de la page précédente)

```
if (!empty($_SERVER['HTTP_USER_AGENT'])){
echo "Votre navigateur semble etre:\n";
echo getBrowser($_SERVER['HTTP_USER_AGENT']);
}

// Test avec des UserAgent connus:
$FF="Mozilla/5.0 (Macintosh; Intel Mac OS X 10.9; rv:32.0) Gecko/
 ↪20100101 Firefox/32.0";
$msie="Mozilla/5.0 (compatible; MSIE 10.6; Windows NT 6.1; Trident/
 ↪5.0; InfoPath.2; SLCC1; .NET CLR 3.0.4506.2152; .NET CLR 3.5.
 ↪30729; .NET CLR 2.0.50727) 3gpp-gba UNTRUSTED/1.0";
$chrome="Mozilla/5.0 (Windows NT 6.3; Win64; x64) AppleWebKit/537.
 ↪36 (KHTML, like Gecko) Chrome/37.0.2049.0 Safari/537.36";
$safari="Mozilla/5.0 (iPad; CPU OS 6_0 like Mac OS X) AppleWebKit/
 ↪536.26 (KHTML, like Gecko) Version/6.0 Mobile/10A5355d Safari/
 ↪8536.25";

echo "<br/> Test Firefox:<br/>\n";
echo getBrowser($FF)."<br/>\n";
echo "<br/> Test MSIE: \n";
echo getBrowser($msie)."<br/>\n";
echo "<br/> Test Chrome: \n";
echo getBrowser($chrome)."<br/>\n";
echo "<br/> Test Safari: \n";
echo getBrowser($safari);
```

Résultat :

```
<br/> Test Firefox:<br/>
Firefox/32.0<br/>
<br/> Test MSIE:
MSIE 10.6<br/>
<br/> Test Chrome:
Chrome/37.<br/>
<br/> Test Safari:
Safari Version/6.0
```

3.10 Le typage en PHP

3.10.1 Les fonctions *gettype( )* et *settype( )*

gettype( ) renvoie l'un des résultats suivants :

— integer

— double
— string
— array
— object
— class
— « unknown type »

settype( ) change le type d'un élément

```
<?php
$a=3.5;
settype($a,"integer");
echo "le contenu de la variable a est ".$a;
```

dont le résultat sera :

```
le contenu de la variable a est 3
```

3.10.2 Fonctions de test

— *is_int()*
— *is_long()*
— *is_double()*
— *is_array()*
— *is_object()*
— *is_string()*

Attention : N'oubliez pas comme en JavaScript la différence entre l'opérateur == et ===

Le premier vérifie l'égalité des contenus en ne tenant pas compte d'une éventuelle différence de typage (int ou string par exemple) tandis que le second vérifie une égalité stricte.

En d'autres termes : 5 == « 5 » est VRAI tandis que 5 === « 5 » est FAUX

3.11 Quelques particularités de PHP

3.11.1 Valeurs des variables :

```
    <?php
$toto = "Bonjour<br/>\n";
$var = "toto";
echo $$var;
```

dont le résultat sera toujours :

3.11.2 Résultat brut

```
Bonjour<br/>
```

3.11.3 La fonction *eval()*

— Permet l'évaluation d'expressions arithmetiques directement en PHP.
— Existe aussi en JavaScript.
— Délicat à manipuler, problématique en termes de sécurité.

3.12 Les tableaux en PHP

3.12.1 Tableaux associatifs - parcours avec boucle foreach :

```
<?php
  $jours=array("Lu"=>"Lundi","Ma"=>"Mardi",
        "Me"=>"Mercredi","Je"=>"Jeudi","Ve"=>"Vendredi",
        "Sa"=>"Samedi","Di"=>"Dimanche" );

  foreach($jours as $key=>$val) echo $key." ".$val."<br>\n";
```

Ce qui donne :

```
Lu Lundi
Ma Mardi
Me Mercredi
Je Jeudi
Ve Vendredi
Sa Samedi
Di Dimanche
```

3.12.2 Affichage avec print_r() :

```
<?php
     print_r($jours);
```

3.12.3 Résultat brut html :

```
Array
(
    [Lu] => Lundi
    [Ma] => Mardi
    [Me] => Mercredi
    [Je] => Jeudi
    [Ve] => Vendredi
    [Sa] => Samedi
    [Di] => Dimanche
)
```

3.12.4 Essayez vous-même

tabs

3.12.5 Utilisation de la fonction array_walk :

```
<?php array_walk($jours,'aff_tab'); ?>
```

En ayant défini au préalable :

```
<?php
function aff_tab($val, $key){
echo "$key-$val<br/>\n";
}
```

On obtient le même résultat qu'avec la boucle foreach

3.12.6 Tri simple d'un tableau :

```
<?php
  sort($jours);
  array_walk($jours,'aff_tab');
```

3.12.7 On obtient :

```
0-Dimanche
1-Jeudi
2-Lundi
3-Mardi
```

(suite sur la page suivante)

(suite de la page précédente)

```
4-Mercredi
5-Samedi
6-Vendredi
```

C'est à dire que :

— Le tableau est trié selon l'ordre de ses valeurs

— les clefs sont effacées et réaffectées avec des entiers.

Si on veut préserver également les clefs du tableau associatif, il faut utiliser la méthode suivante :

3.12.8 Tri selon l'ordre naturel avec natsort

```
<?php
    $jours=array("Lu"=>"Lundi","Ma"=>"Mardi",
    "Me"=>"Mercredi","Je"=>"Jeudi","Ve"=>"Vendredi",
    "Sa"=>"Samedi","Di"=>"Dimanche" );
    var_dump($jours);
    natsort($jours);
    var_dump($jours);
```

3.12.9 Résultat brut html

```
/Users/roza/work/iut/prog/php/source/exemples/tritab.php:5:
array(7) {
  'Lu' =>
  string(5) "Lundi"
  'Ma' =>
  string(5) "Mardi"
  'Me' =>
  string(8) "Mercredi"
  'Je' =>
  string(5) "Jeudi"
  'Ve' =>
  string(8) "Vendredi"
  'Sa' =>
  string(6) "Samedi"
  'Di' =>
  string(8) "Dimanche"
}
/Users/roza/work/iut/prog/php/source/exemples/tritab.php:7:
array(7) {
  'Di' =>
  string(8) "Dimanche"
  'Je' =>
```

(suite sur la page suivante)

(suite de la page précédente)

```
  string(5) "Jeudi"
  'Lu' =>
  string(5) "Lundi"
  'Ma' =>
  string(5) "Mardi"
  'Me' =>
  string(8) "Mercredi"
  'Sa' =>
  string(6) "Samedi"
  'Ve' =>
  string(8) "Vendredi"
}
```

3.12.10 Exécution

tritabnat

On peut aussi utiliser la fonction natcasesort() si on ne veut pas se préoccuper de la casse des chaines présentes dans le tableau, soit à peu près l'ordre du dictionnaire ...

3.13 Les tableaux prédéfinis de PHP : Superglobales

3.13.1 Les Superglobales de PHP

Ce sont des tableaux concernant pour l'essentiel le protocole HTTP ou la gestion de Cookies ou des Sessions.

- $_GET[ ], $_POST[ ] ou $_REQUEST[ ] qui englobe les 2
- $_SERVER[ ] : Variables décrivant le client ou la page courante
- $_GLOBALS[ ] variables globales
- $_COOKIE[ ] pour les cookies
- $_SESSION[ ] pour les sessions

3.13.2 Exemple récupération de $_SERVER[ ] grâce à la fonction getenv() :

```
<?php
function infos(){
  $env = array('remote_addr','http_accept_language','http_
↪host',
  'http_user_agent','script_filename','server_addr',
  'server_name','server_signature','server_software',
  'request_method','query_string','request_uri','script_name
↪');
```

(suite sur la page suivante)

(suite de la page précédente)

```
  // Construction d'un tableau associatif
  // Avec les valeurs lues dans l'environnement
  $retour =array();
  foreach ($env as $clef) $retour[$clef] = getenv($clef);
  return $retour;
}

echo("Voici les infos disponibles:<BR>");
$tab = infos();
foreach ($tab as $clef=>$val) echo $clef." :".$val."<br>\n";
```

3.13.3 Résultat

```
Voici les infos disponibles:
remote_addr :::1
http_accept_language :fr-fr
http_host :localhost
http_user_agent :Mozilla/5.0 (Macintosh; U; Intel Mac OS X␣
 ↪10_6_4; fr-fr)
AppleWebKit/533.18.1 (KHTML, like Gecko) Version/5.0.2␣
 ↪Safari/533.18.5
script_filename :/Users/roza/Sites/php/exemples/infospy.php
server_addr :::1
server_name :localhost
server_signature :
server_software :Apache/2.2.14 (Unix) mod_ssl/2.2.14
OpenSSL/0.9.8l DAV/2 PHP/5.3.2
request_method :GET
query_string :
request_uri :/~roza/php/exemples/infospy.php
script_name :/~roza/php/exemples/infospy.php
`User-Agent <http://localhost/~roza/php/exemples/infospy.
 ↪php>`_
```

3.13.4 Exécution

infospy

3.14 L'inclusion de fichiers externes

3.14.1 *include* :

— Semblable aux *include* du C/C++
— Réalise une inclusion physique du fichier demandé

3.14.2 *include_once* :

— identique au include
— protège contre d'éventuelles inclusions multiples
— qui pourraient mener à des erreurs (redéclarations, etc.)

```php
<?php include_once("connect.php"); ?>
```

3.14.3 *require* et *require_once* :

— fonctionnent comme le include et le include_once respectivement
— mais le programme s'arrête si le fichier inclus n'existe pas

```php
<?php
    require("malib.php");
    require_once("connect.php");
?>
```

3.14.4 *dirname()*

Pour savoir dans quel répertoire on se trouve on peut utiliser la fonction PHP dirname()

```php
<?php
include_once(dirname(__FILE__) . '/config/config.inc.php');
?>
```

Indication : Lorsqu'on inclus ou désigne des fichiers, il vaut mieux utiliser des chemins relatifs pour repérer les fichiers (comme ci dessus) plutôt que de donner un chemin absolu par rapport à la racine du serveur du style /home/user/www/config/config.inc.php Cela sera beaucoup plus portable d'un serveur à l'autre et vous évitera bien des déboires !

Avertissement : L'utilisation systématique de la version avec *once* (include_once ou require_once)n'est pas recommandée car elle peut causer des ralentissements à l'exécution du programme.

3.15 Les Objets en PHP

3.15.1 Evolutions et grands principes :

— Les objets existent en PHP à partir de la version 4
— Changements importants en PHP 5 : Convergence vers le modèle objet de Java
— Introduction comme en Java d'interfaces et de classes abstraites
— emploi des modifieurs *private* et *public* comme en java
— On retrouve aussi **__toString()**, **__clone()** et un mécanisme de traitement des exceptions semblable à celui de Java.
— Les constructeurs s'appellent désormais : **__construct()**
— et les destructeurs **__destruct()**
— les méthodes portent le mot clef *function* mais ne signalent pas leur type de retour
— les commentaires de documentation se font à la manière de Java

---

Indication : Les objets s'instancient avec le mot clef **new** comme en Java ou C++ mais on utilise **->** pour signifier l'appel d'une méthode. Le **.** est déjà pris pour la concaténation des chaînes...

3.15.2 Un Objet Simple Etudiant en PHP

Fabriquons maintenant un objet simple en PHP. Ecrivons un objet représentant un étudiant avec ses données :
— identifiant
— nom
— date de naissance
et des méthodes pour opérer sur ces données :
— constructeur
— getters et setters
— equals()
— toString() pour affichage

ce qui donne le code suivant :

```
<?php
/** Classe Etudiant en PHP */

class Etudiant{
  /** Identification unique d'un etudiant */
  protected $etudiant_id;
  /** Nom de l'etudiant */
  protected $nom;
  /** Date de naissance de l'etudiant */
```

(suite sur la page suivante)

(suite de la page précédente)

```
  protected $naissance;

  public function __construct($id, $nom,$naissance){
    $this->etudiant_id = (int)$id;  // cast vers integer
    $this->nom = (string)$nom;      // cast vers string
    $this->naissance= (int)$naissance;  // cast vers date(timestamp)
  }

  /**
   * Fonction de comparaison simplifiee entre etudiants
   * == comparera id, nom et naissance
   */
  public function equals(etudiant $etudiant){
    return ($this->getId() == $etudiant->getId());
  }

  public function getId(){
    return $this->etudiant_id;
  }

  public function getNom(){
    return $this->nom;
  }

  public function getNaissance(){
    return $this->naissance;
  }

  public function __toString(){
          setlocale(LC_TIME, "fr_FR");
          $ne=strftime('%A %d %B %Y',$this->naissance);
    return 'etudiant: id=' . $this->getId().', nom='.$this->
↪getNom()." $ne";
  }
}
/* Test : */
date_default_timezone_set('Europe/Paris');
$etu=new Etudiant(234,"Talon",time());
var_dump($etu);
echo "<br/>";
echo $etu;
```

Resultat :

```
/Users/roza/work/iut/prog/php/source/exemples/etudiant.
↪php:47:
class Etudiant#1 (3) {
```

(suite sur la page suivante)

(suite de la page précédente)

```
  protected $etudiant_id =>
  int(234)
  protected $nom =>
  string(5) "Talon"
  protected $naissance =>
  int(1591799978)
}
<br/>etudiant: id=234, nom=Talon Mercredi 10 juin 2020
```

Execution

etudiant php

Indication : Lorsqu'on utilise l'opérateur de comparaison **==** sur des objets, ceux-ci sont comparés en utilisant les règles suivantes :

deux objets sont égaux s'ils sont **instances de la même classe et ont les mêmes attributs et valeurs**, les valeurs étant comparées avec l'opérateur ==.

Lors de l'utilisation de l'opérateur d'identité **===**, les objets sont identiques **uniquement s'ils font référence à la même instance de la même classe**.

Avertissement : Beaucoup de programmeurs débutants se contentent d'utiliser PHP comme un langage purement procédural sans utiliser son côté objet. Ceci les bloquera tôt ou tard dans l'apprentissage du langage et des grands frameworks PHP. Si vous êtes trop mal à l'aise avec les Objets, suivez d'abord un cours d'introduction à la programmation Objet, en Java, Python ou C++ par exemple.

3.16 Les collections en PHP

3.16.1 En PHP standard, Collections = Arrays :

— Si on se contente de ce qu'offre PHP en standard, les collections se résument à l'utilisation des tableaux associatifs en PHP
— Le framework des *Collections* en Java est beaucoup plus riche

3.16.2 DataStructures ds :

Il faut installer une librairie supplémentaire *ds* (DataStructures) pour avoir accès à un Framework similaire en PHP.

Voir Documentation Collections php (http ://php.net/manual/fr/book.ds.php)

On va pour utiliser la commande *pecl* pour installer *ds* comme une extension :

```
pecl install ds
```

Puis charger l'extension en ajoutant aux fichiers .ini de PHP :

```
extension=ds.so
```

Cette extension nous donne accès à des classes similaires à celles du framework des Collections en Java. Les classes proposées sont par exemple :

- Sequence
- Vector
- Deque
- Vector
- Pair
- Set
- Stack
- Queue
- PriorityQueue

3.16.3 Utilisation de Ds :

Les classes et interfaces de ds s'utilisent dans un espace de nommage **Ds** :

```
<?php
$vector = new \Ds\Vector();
$vector->push("un");
$vector->push("deux");
$vector->push("trois", "quatre");
// ...[ , ] = unpacking
$vector->push(...["cinq", "six"]);
print_r($vector);
?>
```

3.16.4 Exemple d'utilisation de la classe *Set*

Utilisons maintenant concrètement la classe *Set*.

Question HTML dans un select multiple :

Prenons un petit formulaire en HTML qui propose un choix de couleurs dans un select :

```
<p>Quelles sont les couleurs du maillot des panthères du Fleury␣
↪Loiret Handball ?
  <select name="coul[]" multiple size=5>
```

(suite sur la page suivante)

(suite de la page précédente)

```
      <option value="jaune">Jaune
      <option value="rose">Rose
      <option value="bleu">Bleu
      <option value="noir">Noir
      <option value="blanc">Blanc
      <option value="vert">Vert
  </select>
</p>
```

Réponse PHP avec Collections

```
<?php
if (!empty($_GET['coul'])){
    $couleursReponse = new \Ds\Set($_GET['coul']);
    $couleursCorrectes = new \Ds\Set(['rose','noir','blanc
↪']);
// Calculons la différence des 2 ensembles et voyons si␣
↪elle est vide
if (($couleursReponse->diff($couleursCorrectes))->isEmpty())
        echo "Bravo les couleurs de Fleury Hand Ball sont␣
↪bien Rose Noir et Blanc !";
else
        echo "Mauvaise réponse : les couleurs de Fleury␣
↪Hand Ball sont: Rose Noir et Blanc !";
}
```

Sans Collections, on utilise les tableaux . . .

Sans **Ds**, nous aurions été obligés de nous contenter de tableaux PHP et d'utiliser par exemple la méthode **array_diff** : Voir : array_diff php (http ://php.net/manual/fr/function.array-diff.php)

Avec des Arrays :

On peut ici s'en sortir avec de simples tableaux PHP en vérifiant que la taille du tableau des réponses données est la même que celle du tableau des bonnes réponses, puis que les contenus de ces tableaux sont identiques.

```
<?php
if (!empty($_GET['coul'])){
    $couleursReponse = $_GET['coul'];
    $couleursCorrectes = array(['rose','noir','blanc']);
    if (count($couleursReponse) == count($couleursCorrectes) && !
↪array_diff($couleursReponse, $couleursCorrectes))
        echo "Bravo les couleurs de Fleury Hand Ball sont bien Rose␣
↪Noir et Blanc !";
```

(suite sur la page suivante)

(suite de la page précédente)

```
else
        echo "Mauvaise réponse : les couleurs de Fleury Hand Ball␣
↪sont: Rose Noir et Blanc !";
}
```

3.17 Connexion aux bases de données depuis PHP avec PDO

3.17.1 Une table simple en SQL :

```
        CREATE TABLE `CARNET` (
          `ID` int(11) NOT NULL AUTO_INCREMENT,
          `NOM` varchar(30) DEFAULT NULL,
          `PRENOM` varchar(30) DEFAULT NULL,
          `NAISSANCE` date DEFAULT NULL,
          `VILLE` varchar(30) DEFAULT NULL,
          PRIMARY KEY (`ID`)
        ) ENGINE=InnoDB AUTO_INCREMENT=13 DEFAULT CHARSET=utf8;

        INSERT INTO `CARNET` VALUES
        (1,'SMITH','JOHN','1980-12-17','ORLEANS'),
        (2,'DURAND','JEAN','1983-01-13','ORLEANS'),
        (3,'GUDULE','JEANNE','1967-11-06','TOURS'),
        (4,'ZAPATA','EMILIO','1956-12-01','ORLEANS'),
        (5,'JOURDAIN','NICOLAS','2000-09-10','TOURS'),
        (6,'DUPUY','MARIE','1986-01-11','BLOIS'),
        (7,'ANDREAS','LOU','1861-02-12','ST Petersbourg'),
        (9,'Kafka','Franz','1883-07-03','Prague'),
        (11,'Dalton','Joe','2003-12-06','Dallas');
```

On insère cette table dans MySQL en ligne de commande ou à l'aide de PHPMyAdmin. Puis, pour consulter cette table depuis PHP, on utilise le connecteur PDO qui offre une interface de connexion utilisable pour tous les SGBD (Systemes de Gestion de Bases de Donnees) habituels comme MySQL, PostgreSQL, Oracle ou Microsoft SQL Server.

3.17.2 Connexion Simple en PHP avec PDO

```
<!doctype html>
<html>
<head>
<title>
Connexion à MySQL avec PDO
</title>
```

(suite sur la page suivante)

(suite de la page précédente)

```
<meta charset="utf-8">
</head>
<body>
<h1>
Interrogation de la table CARNET avec PDO
</h1>

<?php
require("connect.php");
// pour oracle: $dsn="oci:dbname=//serveur:1521/base
// pour sqlite: $dsn="sqlite:/tmp/base.sqlite"
$dsn="mysql:dbname=".BASE.";host=".SERVER;
    try{
      $connexion=new PDO($dsn,USER,PASSWD);
    }
    catch(PDOException $e){
      printf("Échec de la connexion : %s\n", $e->getMessage());
      exit();
    }
$sql="SELECT * from CARNET";
if(!$connexion->query($sql)) echo "Pb d'accès au CARNET";
else{
     foreach ($connexion->query($sql) as $row)
     echo $row['PRENOM']." ".
          $row['NOM']."né(e) le ".
          $row['NAISSANCE']."<br/>\n";
}
?>
</body>
</html>
```

Avec un fichier connect.php contenant les informations de connexion au serveur MySQL :

```
<?php
define('USER',"scott");
define('PASSWD',"tiger");
define('SERVER',"localhost");
define('BASE',"dbscott");
?>
```

Resultat brut html :

```
<!doctype html>
<html>
<head>
<title>
Connexion à MySQL avec PDO
```

(suite sur la page suivante)

(suite de la page précédente)

```
</title>
<meta charset="utf-8">
</head>
<body>
<h1>
Interrogation de la table CARNET avec PDO
</h1>


JOHN SMITHné(e) le 1980-12-17<br/>
JEAN DURANDné(e) le 1983-01-13<br/>
JEANNE GUDULEné(e) le 1967-11-06<br/>
EMILIO ZAPATAné(e) le 1956-12-01<br/>
NICOLAS JOURDAINné(e) le 2000-09-10<br/>
MARIE DUPUYné(e) le 1986-01-11<br/>
LOU ANDREAné(e) le 1900-01-01<br/>
bbb aaané(e) le 2020-04-22<br/>
</body>
</html>
```

Execution

carnet php

Fabrication d'une table HTML avec les résultats :

```
<html>
<head>
<title>
Connexion à MySQL avec PDO
</title>
<meta charset="utf-8">
<body>
<h1>
Interrogation de la table CARNET avec PDO
</h1>

<?php
require("connect.php");
$dsn="mysql:dbname=".BASE.";host=".SERVER;
    try{
      $connexion=new PDO($dsn,USER,PASSWD);
    }
    catch(PDOException $e){
      printf("Échec de la connexion : %s\n", $e->getMessage());
      exit();
```

(suite sur la page suivante)

(suite de la page précédente)

```
    }
$sql="SELECT * from CARNET";
if(!$connexion->query($sql)) echo "Pb d'accès au CARNET";
else{
?>
<table>
<tr> <td> Nom </td> <td> Prénom </td></tr>
  <?php
  foreach ($connexion->query($sql) as $row)
echo "<tr><td>".$row['NOM']."</td><td>".
  $row['PRENOM']."</td></tr>\n";
  ?>
</table>
<?php } ?>
</body>
</html>
```

Résultats bruts :

```
<html>
<head>
<title>
Connexion à MySQL avec PDO
</title>
<meta charset="utf-8">
<body>
<h1>
Interrogation de la table CARNET avec PDO
</h1>


<table>
<tr> <td> Nom </td> <td> Prénom </td></tr>
  <tr><td>SMITH</td><td>JOHN</td></tr>
<tr><td>DURAND</td><td>JEAN</td></tr>
<tr><td>GUDULE</td><td>JEANNE</td></tr>
<tr><td>ZAPATA</td><td>EMILIO</td></tr>
<tr><td>JOURDAIN</td><td>NICOLAS</td></tr>
<tr><td>DUPUY</td><td>MARIE</td></tr>
<tr><td>ANDREA</td><td>LOU</td></tr>
<tr><td>aaa</td><td>bbb</td></tr>
</table>
</body>
</html>
```

Execution

Carnet Table

On peut faire un petit refactoring avec une fonction qui établit la connexion à la base :

```
<?php

require("connect.php");
function connect_bd(){
        $dsn="mysql:dbname=".BASE.";host=".SERVER;
                try{
                $connexion=new PDO($dsn,USER,PASSWD);
                }
                catch(PDOException $e){
                printf("Échec de la connexion : %s\n", $e->
 ↪getMessage());
                exit();
                }
        return $connexion;
}

?>
```

et améliorer l'affichage des résultats :

```
<html>
<head>
<title>
Connexion à MySQL avec PDO
</title>
<meta charset="utf-8">
<link rel="stylesheet" href="tabstyle.css" />
</head>
<body>
<h1>
Interrogation de la table CARNET avec PDO
</h1>

<?php
require_once('connexion.php');
$connexion=connect_bd();
$sql="SELECT * from CARNET";
if(!$connexion->query($sql)) echo "Pb d'accès au CARNET";
else{
?>
<table class="centre" id="jolie">
<tr> <td> ID </td> <td> Prénom </td> <td> Nom </td><td> Naissance </
 ↪td> </tr>
```

(suite sur la page suivante)

(suite de la page précédente)

```
  <?php
  foreach ($connexion->query($sql) as $row)
echo "<tr><td>".$row['ID']."</td>
          <td>".$row['PRENOM']."</td>
          <td>".$row['NOM']."</td>
          <td>".$row['NAISSANCE']."</td></tr><br/>\n";
  ?>
</table>
<?php } ?>
</body>
</html>
```

Avec le fichier CSS :

```
/* Bordure simple autour des tableaux */
table,th, td { border: 1px solid grey;}
table{border-collapse:collapse;}
/* Centrage tableau */
table.centre{ margin:auto;}
/* centrage du texte dans les cellules du tableau */
table.centre td{text-align:center;}

table#jolie tr:first-child{
        background:LightPink;
}
table#jolie tr:nth-child(2n){
        background:#EFD3C9;
}
table#jolie tr:nth-child(2n+3){
        background:#BCBCD0;
}
/* si un tableau a une seule ligne on l'affiche en rouge */
table tr:only-child{
        background:red;
}
```

Résultats bruts :

```
<html>
<head>
<title>
Connexion à MySQL avec PDO
</title>
<meta charset="utf-8">
<link rel="stylesheet" href="tabstyle.css" />
</head>
```

(suite sur la page suivante)

(suite de la page précédente)

```
<body>
<h1>
Interrogation de la table CARNET avec PDO
</h1>


<table class="centre" id="jolie">
<tr> <td> ID </td> <td> Prénom </td> <td> Nom </td><td> Naissance </
↪td> </tr>
  PHP Notice:  Undefined index: ID in /Users/roza/work/iut/prog/php/
↪source/exemples/pdo/cartable2.php on line 25
PHP Stack trace:
PHP   1. {main}() /Users/roza/work/iut/prog/php/source/exemples/pdo/
↪cartable2.php:0

Notice: Undefined index: ID in /Users/roza/work/iut/prog/php/source/
↪exemples/pdo/cartable2.php on line 25

Call Stack:
    0.0002     398656   1. {main}() /Users/roza/work/iut/prog/php/
↪source/exemples/pdo/cartable2.php:0

<tr><td></td>
          <td>JOHN</td>
          <td>SMITH</td>
          <td>1980-12-17</td></tr><br/>
PHP Notice:  Undefined index: ID in /Users/roza/work/iut/prog/php/
↪source/exemples/pdo/cartable2.php on line 25
PHP Stack trace:
PHP   1. {main}() /Users/roza/work/iut/prog/php/source/exemples/pdo/
↪cartable2.php:0

Notice: Undefined index: ID in /Users/roza/work/iut/prog/php/source/
↪exemples/pdo/cartable2.php on line 25

Call Stack:
    0.0002     398656   1. {main}() /Users/roza/work/iut/prog/php/
↪source/exemples/pdo/cartable2.php:0

<tr><td></td>
          <td>JEAN</td>
          <td>DURAND</td>
          <td>1983-01-13</td></tr><br/>
PHP Notice:  Undefined index: ID in /Users/roza/work/iut/prog/php/
↪source/exemples/pdo/cartable2.php on line 25
PHP Stack trace:
PHP   1. {main}() /Users/roza/work/iut/prog/php/source/exemples/pdo/
↪cartable2.php:0
```

(suite sur la page suivante)

(suite de la page précédente)

```
Notice: Undefined index: ID in /Users/roza/work/iut/prog/php/source/
 ↪exemples/pdo/cartable2.php on line 25

Call Stack:
    0.0002     398656   1. {main}() /Users/roza/work/iut/prog/php/
 ↪source/exemples/pdo/cartable2.php:0

<tr><td></td>
          <td>JEANNE</td>
          <td>GUDULE</td>
          <td>1967-11-06</td></tr><br/>
PHP Notice:  Undefined index: ID in /Users/roza/work/iut/prog/php/
 ↪source/exemples/pdo/cartable2.php on line 25
PHP Stack trace:
PHP   1. {main}() /Users/roza/work/iut/prog/php/source/exemples/pdo/
 ↪cartable2.php:0

Notice: Undefined index: ID in /Users/roza/work/iut/prog/php/source/
 ↪exemples/pdo/cartable2.php on line 25

Call Stack:
    0.0002     398656   1. {main}() /Users/roza/work/iut/prog/php/
 ↪source/exemples/pdo/cartable2.php:0

<tr><td></td>
          <td>EMILIO</td>
          <td>ZAPATA</td>
          <td>1956-12-01</td></tr><br/>
PHP Notice:  Undefined index: ID in /Users/roza/work/iut/prog/php/
 ↪source/exemples/pdo/cartable2.php on line 25
PHP Stack trace:
PHP   1. {main}() /Users/roza/work/iut/prog/php/source/exemples/pdo/
 ↪cartable2.php:0

Notice: Undefined index: ID in /Users/roza/work/iut/prog/php/source/
 ↪exemples/pdo/cartable2.php on line 25

Call Stack:
    0.0002     398656   1. {main}() /Users/roza/work/iut/prog/php/
 ↪source/exemples/pdo/cartable2.php:0

<tr><td></td>
          <td>NICOLAS</td>
          <td>JOURDAIN</td>
          <td>2000-09-10</td></tr><br/>
PHP Notice:  Undefined index: ID in /Users/roza/work/iut/prog/php/
 ↪source/exemples/pdo/cartable2.php on line 25
PHP Stack trace:
PHP   1. {main}() /Users/roza/work/iut/prog/php/source/exemples/pdo/
 ↪cartable2.php:0
```

(suite sur la page suivante)

(suite de la page précédente)

```
Notice: Undefined index: ID in /Users/roza/work/iut/prog/php/source/
 ↪exemples/pdo/cartable2.php on line 25

Call Stack:
    0.0002     398656   1. {main}() /Users/roza/work/iut/prog/php/
 ↪source/exemples/pdo/cartable2.php:0

<tr><td></td>
          <td>MARIE</td>
          <td>DUPUY</td>
          <td>1986-01-11</td></tr><br/>
PHP Notice:  Undefined index: ID in /Users/roza/work/iut/prog/php/
 ↪source/exemples/pdo/cartable2.php on line 25
PHP Stack trace:
PHP   1. {main}() /Users/roza/work/iut/prog/php/source/exemples/pdo/
 ↪cartable2.php:0

Notice: Undefined index: ID in /Users/roza/work/iut/prog/php/source/
 ↪exemples/pdo/cartable2.php on line 25

Call Stack:
    0.0002     398656   1. {main}() /Users/roza/work/iut/prog/php/
 ↪source/exemples/pdo/cartable2.php:0

<tr><td></td>
          <td>LOU</td>
          <td>ANDREA</td>
          <td>1900-01-01</td></tr><br/>
PHP Notice:  Undefined index: ID in /Users/roza/work/iut/prog/php/
 ↪source/exemples/pdo/cartable2.php on line 25
PHP Stack trace:
PHP   1. {main}() /Users/roza/work/iut/prog/php/source/exemples/pdo/
 ↪cartable2.php:0

Notice: Undefined index: ID in /Users/roza/work/iut/prog/php/source/
 ↪exemples/pdo/cartable2.php on line 25

Call Stack:
    0.0002     398656   1. {main}() /Users/roza/work/iut/prog/php/
 ↪source/exemples/pdo/cartable2.php:0

<tr><td></td>
          <td>bbb</td>
          <td>aaa</td>
          <td>2020-04-22</td></tr><br/>
</table>
</body>
</html>
```

Execution

Carnet Table Version2

On peut générer des pages différentes avec des listes déroulantes ou des listes de liens, listes à puces etc.

Création d'une liste déroulante :

```
<!doctype html>
<html>
<head>
<title>
Connexion à MySQL avec PDO
</title>
<meta charset="utf-8">
</head>
<body>
<h1>
Interrogation de la table CARNET avec PDO
</h1>
<?php
require_once('connexion.php');
$connexion=connect_bd();
$sql="SELECT * from CARNET";
if(!$connexion->query($sql)) echo "Pb d'accès au CARNET";
else {
      ?>
      <form action="recherche.php" method="GET">
             <select name="ID">
       <?php
       foreach ($connexion->query($sql) as $row)
       if(!empty($row['NOM']))
             echo "<option value='".$row['ID']."'>"
         .$row['PRENOM']." ".$row['NOM']."</option>\n";
       ?>
      </select>
      <input type="submit" value="Rechercher">
      </form>
      <?php
       }
      ?>
</body>
</html>
```

Remarquez l'usage de la clef primaire de la table comme value des options, ce qui assurent l'unicité des valeurs et évite toute amiguité.

Résultats bruts :

```
<!doctype html>
<html>
<head>
<title>
Connexion à MySQL avec PDO
</title>
<meta charset="utf-8">
</head>
<body>
<h1>
Interrogation de la table CARNET avec PDO
</h1>

     <form action="recherche.php" method="GET">
            <select name="ID">
      PHP Notice:  Undefined index: ID in /Users/roza/work/iut/prog/
 ↪php/source/exemples/pdo/carselect.php on line 25
PHP Stack trace:
PHP   1. {main}() /Users/roza/work/iut/prog/php/source/exemples/pdo/
 ↪carselect.php:0

Notice: Undefined index: ID in /Users/roza/work/iut/prog/php/source/
 ↪exemples/pdo/carselect.php on line 25

Call Stack:
    0.0002     398576   1. {main}() /Users/roza/work/iut/prog/php/
 ↪source/exemples/pdo/carselect.php:0

<option value=''>JOHN SMITH</option>
PHP Notice:  Undefined index: ID in /Users/roza/work/iut/prog/php/
 ↪source/exemples/pdo/carselect.php on line 25
PHP Stack trace:
PHP   1. {main}() /Users/roza/work/iut/prog/php/source/exemples/pdo/
 ↪carselect.php:0

Notice: Undefined index: ID in /Users/roza/work/iut/prog/php/source/
 ↪exemples/pdo/carselect.php on line 25

Call Stack:
    0.0002     398576   1. {main}() /Users/roza/work/iut/prog/php/
 ↪source/exemples/pdo/carselect.php:0

<option value=''>JEAN DURAND</option>
PHP Notice:  Undefined index: ID in /Users/roza/work/iut/prog/php/
 ↪source/exemples/pdo/carselect.php on line 25
PHP Stack trace:
PHP   1. {main}() /Users/roza/work/iut/prog/php/source/exemples/pdo/
 ↪carselect.php:0
```

(suite sur la page suivante)

(suite de la page précédente)

```
Notice: Undefined index: ID in /Users/roza/work/iut/prog/php/source/
 ↪exemples/pdo/carselect.php on line 25

Call Stack:
    0.0002     398576   1. {main}() /Users/roza/work/iut/prog/php/
 ↪source/exemples/pdo/carselect.php:0

<option value=''>JEANNE GUDULE</option>
PHP Notice:  Undefined index: ID in /Users/roza/work/iut/prog/php/
 ↪source/exemples/pdo/carselect.php on line 25
PHP Stack trace:
PHP   1. {main}() /Users/roza/work/iut/prog/php/source/exemples/pdo/
 ↪carselect.php:0

Notice: Undefined index: ID in /Users/roza/work/iut/prog/php/source/
 ↪exemples/pdo/carselect.php on line 25

Call Stack:
    0.0002     398576   1. {main}() /Users/roza/work/iut/prog/php/
 ↪source/exemples/pdo/carselect.php:0

<option value=''>EMILIO ZAPATA</option>
PHP Notice:  Undefined index: ID in /Users/roza/work/iut/prog/php/
 ↪source/exemples/pdo/carselect.php on line 25
PHP Stack trace:
PHP   1. {main}() /Users/roza/work/iut/prog/php/source/exemples/pdo/
 ↪carselect.php:0

Notice: Undefined index: ID in /Users/roza/work/iut/prog/php/source/
 ↪exemples/pdo/carselect.php on line 25

Call Stack:
    0.0002     398576   1. {main}() /Users/roza/work/iut/prog/php/
 ↪source/exemples/pdo/carselect.php:0

<option value=''>NICOLAS JOURDAIN</option>
PHP Notice:  Undefined index: ID in /Users/roza/work/iut/prog/php/
 ↪source/exemples/pdo/carselect.php on line 25
PHP Stack trace:
PHP   1. {main}() /Users/roza/work/iut/prog/php/source/exemples/pdo/
 ↪carselect.php:0

Notice: Undefined index: ID in /Users/roza/work/iut/prog/php/source/
 ↪exemples/pdo/carselect.php on line 25

Call Stack:
    0.0002     398576   1. {main}() /Users/roza/work/iut/prog/php/
 ↪source/exemples/pdo/carselect.php:0
```

(suite sur la page suivante)

(suite de la page précédente)

```
<option value=''>MARIE DUPUY</option>
PHP Notice:  Undefined index: ID in /Users/roza/work/iut/prog/php/
 ↪source/exemples/pdo/carselect.php on line 25
PHP Stack trace:
PHP   1. {main}() /Users/roza/work/iut/prog/php/source/exemples/pdo/
 ↪carselect.php:0

Notice: Undefined index: ID in /Users/roza/work/iut/prog/php/source/
 ↪exemples/pdo/carselect.php on line 25

Call Stack:
    0.0002     398576   1. {main}() /Users/roza/work/iut/prog/php/
 ↪source/exemples/pdo/carselect.php:0

<option value=''>LOU ANDREA</option>
PHP Notice:  Undefined index: ID in /Users/roza/work/iut/prog/php/
 ↪source/exemples/pdo/carselect.php on line 25
PHP Stack trace:
PHP   1. {main}() /Users/roza/work/iut/prog/php/source/exemples/pdo/
 ↪carselect.php:0

Notice: Undefined index: ID in /Users/roza/work/iut/prog/php/source/
 ↪exemples/pdo/carselect.php on line 25

Call Stack:
    0.0002     398576   1. {main}() /Users/roza/work/iut/prog/php/
 ↪source/exemples/pdo/carselect.php:0

<option value=''>bbb aaa</option>
     </select>
     <input type="submit" value="Rechercher">
     </form>
     </body>
</html>
```

Execution

Carnet Select

3.18 Requêtes préparées en PHP

3.18.1 Recherche simple en PHP avec PDO

```
<!doctype html>
<html>
<head>
<title>
Recherche d'une personne par ID
</title>
<meta charset="utf-8">
</head>
<body>
<?php $wanted=$_GET['ID'];
if (!empty($wanted)){
        echo "<h1>Recherche de  $wanted </h1>";
        require_once('connexion.php');
        $connexion=connect_bd();
        $sql="SELECT * from CARNET where ID='".$wanted."'";
        if(!$connexion->query($sql))
                echo "Pb de requete";
        else{
        foreach ($connexion->query($sql) as $row)
                echo $row['NOM']." ".$row['PRENOM']."</br>\n";
  }
}
  ?>
</body>
</html>
```

Appel avec le paramètre ID passé sur l'URL : php exemples/pdo/recherche.php ?ID=3

Execution

recherche php

Mais lorsqu'il y a de nombreux paramètres, on se retrouve avec de nombreuses concaténations de chaines entourées de "cotes" ce qui est une grande source d'erreurs et de lenteurs d'écriture. Pour remédier à cela, on peut utiliser des requêtes préparées qui permettent de bien dissocier la requête des paramètres qui vont lui être fournis avant son exécution. Ces *PreparedStatement* seront également bien préférables en termes de sécurité et à utiliser **systématiquement**.

3.18.2 Recherche avec PreparedStatement

```
<!doctype html>
<html>
<head>
<title>
Recherche d'une personne par ID
</title>
<meta charset="utf-8">
</head>
<body>
<?php $wanted=$_GET['ID'];
if (!empty($wanted)){
  echo "<h1>Recherche de  $wanted </h1>";
  require_once('connexion.php');
  $connexion=connect_bd();
$sql="SELECT * from CARNET where ID=:id";
$stmt=$connexion->prepare($sql);
$stmt->bindParam(':id', $_GET['ID']);
$stmt->execute();
if (!$stmt) echo "Pb d'accès au CARNET";
else{
    if ($stmt->rowCount()==0) echo "Inconnu !<br/>";
    else
     foreach ($stmt as $row)
     echo $row['PRENOM']." ".$row['NOM'].
          "né(e) le ".$row['NAISSANCE']."<br/>";
}
}
?>
</body>
</html>
```

Les requêtes préparées limitent fortement la possibilité d'*injections SQL* comme nous le verront plus tard.

3.19 Compléments sur PDO - Sécurité

3.19.1 Filtrage d'entrées

On peut vouloir nourrir une requête directement avec des données provenant d'un formulaire :

```
<?php
  $sql = sprintf(
    'SELECT id FROM CARNET WHERE email = "%s"',$_GET['email'])
  );
```

On voit bien que la valeur de l'entrée email dans le tableau _GET n'est absolument pas vérfifiée

avant son utilisation !

On peut essayer dans ce cas d'utiliser un filtre PHP pour contrôler un peu les choses :

```
<?php
  $sql = sprintf(
      'SELECT id FROM CARNET WHERE email = "%s"',
      filter_input(INPUT_GET, 'email')
  );
```

Mais ceci constitue une manière peu sûre de procéder malgré l'utilisation du filtre PHP. Cela laisse en effet la possiblilité d'insertion de code malveillant non contrôlé.

L'exemple classique est la requête SQL construite dans la chaine suivante :

```
<?php
  $sql = "SELECT nom FROM USERS WHERE login='".
         $_REQUEST['login']."' AND PASSWD='".
         $_REQUEST['pass']."'";
```

Qui donne lors de son exécution avec $_REQUEST[“login”] = » toto” – » :

```
SELECT nom FROM USERS WHERE login='toto'  -- ' AND PASSWD= '".$_
↪REQUEST['pass']."' ";
```

Avertissement : Le - - constituant un début de commentaire SQL, ceci constitue une injection SQL qui est l'une des principales failles de sécurité exploitées par les Hackers. Pour s'en prémunir, il faut utiliser **à la fois** le filtrage des entrées et les requêtes préparées.

```
<?php
$sql = 'SELECT id FROM CARNET WHERE email = :email';
$statement = $pdo->prepare($sql);
$email = filter_input(INPUT_GET, 'email');
$statement->bindValue(':email', $email);
```

Il faut parfois préciser dans un troisième argument le type des paramètres attendus :

```
<?php
  $sql = 'SELECT email FROM CARNET WHERE id = :id';
  $statement = $pdo->prepare($sql);
  $userId = filter_input(INPUT_GET, 'id');
  $statement->bindValue(':id', $userId, PDO::PARAM_INT);
```

3.20 Filtrage en PHP

Les vérifications dans les formulaires HTML5 et en JavaScript sont valables uniquement côté client. Pour des raisons de sécurité, il faut réeffectuer complètement toutes les vérifications côté

serveur. PHP met pour cela à la disposition des programmeurs tout un ensemble de filtres. La première des vérifications consiste à bien prendre en compte les caractères spéciaux.

3.20.1 Gestion des caractères spéciaux :

Les « magic quotes » :

Avant PHP 5.4, une directive concernant ces magic quotes existait. Si dans le fichier de configuration de PHP, la directive magic_quotes_gpc etait activée, PHP modifiait automatiquement certaines entrées de formulaires en procédant à des protections de certains caractères spéciaux par des insertion de « backslashes ». Par exemple

- les caractères accentués
- les apostrophes
- les backslashes

Un peu à la manière de la fonction addslashes().

Cette protection était destinée à préparer les données avant des requêtes SQL pour empêcher une éventuelle injection SQL. Ce comportement automatique est toutefois parfois gênant si on veut simplement réafficher les chaines saisies et non pas les utiliser dans des requêtes SQL. En outre, on ne veut pas toujours protéger les chaines de la même façon selon l'usage qu'on veut en faire par la suite.

On peut vouloir dans certains cas, protéger des chaines par exemple :

- htmlspecialchars() pour éviter des injections de code HTML
- PDO : :quote() pour se protéger d'injections SQL

```
<?php
  $pdo = new PDO('sqlite:./tmp/mydb.sqlite');
  $string = 'Orléans';
  print "Chaîne sans quotes: $string\n";
  print "Chaîne avec quotes: " . $pdo->quote($string) . "\n";
```

Cela ne vous dispense pas d'utiliser les *PreparedStatement* vus précédemment.

Les filtres PHP :

Les plus directs à utiliser sur les formulaires sont basés sur la fonction filter_input() avec en paramètre INPUT_GET ou INPUT_POST Voici quelques exemples typiques :

```
<?php
    $entier = filter_input(INPUT_POST, 'var1', FILTER_SANITIZE_
↪NUMBER_INT);

    $url = filter_input(INPUT_POST, 'var2', FILTER_VALIDATE_URL);

    if (!filter_input(INPUT_GET, "email",  FILTER_VALIDATE_EMAIL))
        echo("Email non valide");
```

(suite sur la page suivante)

(suite de la page précédente)

```
    else echo("Email valide");
    ?>
```

On peut aussi filtrer des carctères spéciaux :

```
<?php
$search_html = filter_input(INPUT_GET, 'search', FILTER_SANITIZE_
↪SPECIAL_CHARS);
$search_url = filter_input(INPUT_GET, 'search', FILTER_SANITIZE_
↪ENCODED);
echo "Vous avez recherché $search_html.\n";
echo "<a href='?search=$search_url'>Nouvelle recherche.</a>";
?>
```

3.21 Gestion des transactions avec PDO

3.21.1 Problème

Une série de requêtes SQL sont logiquement liées entre elles et on voudrait qu'elles soient toutes exécutées ou aucune. En effet dans certains cas, la prise en compte d'une partie des requêtes seulement peut conduire à une incohérence dans le système d'information. La base de données peut ainsi être corrompue et très difficile à rectifier par la suite. Par exemple, si on a 2 requêtes qui se suivent et qui sont liées :

```
<?php
  require 'connexion.php';
  $connexion=connect_bd();

  $stmt1 = $pdo->prepare('
      UPDATE compte
      SET solde = solde - :montant
      WHERE nom = :nom
  ');
  $stmt2 = $pdo->prepare('
      UPDATE compte
      SET solde = solde + :montant
      WHERE nom = :nom
  ');

  // Retrait du Compte1
  $cpte1 = 'Compte1';
  $montant = 50;
  $stmt1->bindParam(':nom', $cpte1);
  $stmt1->bindParam(':solde', $montant, PDO::PARAM_INT);
  $stmt1->execute();
```

(suite sur la page suivante)

(suite de la page précédente)

```
  // Credit du Compte2
  $cpte2 = 'Compte2';
  $depot = 50;
  $stmt2->bindParam(':nom', $cpte2);
  $stmt->bindParam(':montant', $depot, PDO::PARAM_INT);
  $stmt->execute();
?>
```

Ceci peut conduire à un problème en cas d'interruption de cette séquence. En particulier le Compte1 peut avoir été débité sans que le Compte2 soit crédité.

On peut résoudre cette fragilité en utilisant une transaction :

```
<?php
  require 'connexion.php';
  $connexion=connect_bd();

  $stmt1 = $connexion->prepare('
      UPDATE compte
      SET solde = solde - :solde
      WHERE nom = :nom
  ');
  $stmt2 = $connexion->prepare('
      UPDATE compte
      SET solde = solde + :montant
      WHERE nom = :nom
  ');

  // On commence la transaction
  $connexion ->beginTransaction()
  // Retrait du Compte1
  $cpte1 = 'Compte1';
  $montant = 100;
  $stmt1->bindParam(':nom', $cpte1);
  $stmt1->bindParam(':solde', $montant, PDO::PARAM_INT);
  $stmt1->execute();

  // Credit du Compte2
  $cpte2 = 'Compte2';
  $depot = 50;
  $stmt2->bindParam(':nom', $cpte2);
  $stmt->bindParam(':montant', $depot, PDO::PARAM_INT);
  $stmt->execute();
  //on termine la transaction
  $connexion -> commit();
?>
```

Si la séquence échoue, PDO commandera un *RollBack* automatique, c'est à dire une annulation de toute la transaction.

3.22 Connexion persistante avec PDO

Pour créer une connexion persistante avec PDO, il suffit d'utiliser l'attribut *ATTR PERSISTENT* à l'instanciation de l'objet PDO. Lors des appels ultérieurs, si les paramètres de création sont identiques, l'objet déjà créé sera simplement réutilisé.

```
<?php
function connect_db()
{
  $dsn="mysql:dbname=".BASE.";host=".SERVER;
  try
  {
      $connexion=new PDO($dsn,USER,PASSWD,
          array(PDO::ATTR_PERSISTENT =>true));
  }
  catch(PDOException $e)
  {
      printf("Échec de la connexion : %s\n", $e->getMessage());
      exit();
  }
  return $connexion;
}
?>
```

3.23 Validation et contrôle d'entrées avec GUMP

On peut également utiliser des librairies aidant à gérer la validation comme : GUMP (https ://github.com/Wixel/GUMP)

3.23.1 Pour installer :

Créons un fichier composer.json dans notre répertoire de travail :

```
{
  "require": {
      "wixel/gump": "dev-master"
   }
}
```

Puis utilisons l'outil PHP `composer` pour installer les composants demandés :

```
composer update
```

3.23.2 Nous pouvons utiliser Gump pour vérifier des entrées :

```php
<?php
require "vendor/wixel/gump/gump.class.php";

$is_valid = GUMP::is_valid($_POST, array(
    'username' => 'required|alpha_numeric',
    'password' => 'required|max_len,100|min_len,6'
));

if($is_valid) {
        echo "valid username and password ";
} else {
    print_r($is_valid);
}

?>
```

3.23.3 ou de manière plus détaillée :

```php
<?php
require "vendor/wixel/gump/gump.class.php";

if ($_SERVER["REQUEST_METHOD"] == "POST"){
  $validator = new GUMP();

  $name = $_POST['name'];
  $password = $_POST['password'];

  $_POST = array(
    'name' => $name,
    'password' => $password);

  // néttoie les données et convertit les chaines en utf-8 si besoin
  $_POST = $validator->sanitize($_POST);

  // Définissons les règles et les filtres:
  $rules = array(
    'name' => 'required|alpha_numeric|max_len,100|min_len,6',
    'password' => 'required|max_len,100|min_len,6');

  $filters = array(
    'name'    => 'trim|sanitize_string',
    'password'    => 'trim|base64_encode');

  // On applique les filtres
  $_POST = $validator->filter($_POST, $filters);
```

(suite sur la page suivante)

(suite de la page précédente)

```
  // On valide
  $is_valid  = $validator->validate($_POST, $rules);

  // On vérifie le résultat
  if ($is_valid === true )
  {
    echo $name;
    echo $password;
    exit;
  }
  else
  {
    echo "Erreurs détectées dans les entrées:\n";
    // on affiche les erreurs en html
    echo $validator->get_readable_errors(true);
  }
}
```

3.24 Http et la conservation d'informations sur les clients

3.24.1 Problème

Le protocole HTTP est un protocole *sans mémoire*. Une requête d'un client ou les informations qui y sont rattachées ne sont pas mémorisées par défaut.

Plusieurs techniques ont été développées pour remédier à ce manque :

- Envoyer de l'information sur l'URL
- Utiliser un champ caché HTML
- Utiliser des Cookies
- Utiliser des Sessions

Envoi d'information sur l'URL :

Considérons une première page, page1.php :

```
<!doctype html>
<html lang="fr">
<head>
<meta charset="utf-8"/>
<title> Formulaires</title>
</head>
<body>
<?php
        if (!isset($_GET['login'])){
```

(suite sur la page suivante)

(suite de la page précédente)

```
                ?>
        <form method='GET'
                action=<?php echo $_SERVER['PHP_SELF'] ?>
        >
        <p>Login: <input type="text" name="login"></p>
        <input type="submit" value="Valider">
        </form>
        <?php
        }
        else {
        header('Location:page2.php?login='.$_GET['login']);
        }
?>
</body>
</html>
```

qui se poursuit par une page2 :

```
<!doctype html>
<html lang="fr">
<head>
<meta charset="utf-8"/>
<title>Formulaires</title>
</head>
<body>
<?php
        if (isset($_GET['login'])){
                echo $_GET['login'];
        }
        else {
        header('Location:page1.php');
        }
?>
</body>
</html>
```

Exécution :

Passage d'info sur l'URL

Utiliser des cookies :

L'information est stockée dans un petit fichier texte sur le navigateur du client. On peut par exemple s'en servir pour stocker un identifiant de session, un login, un compteur de visites ou encore mesurer un temps de connexion.

```php
<?php
if (isset($_COOKIE['compteur']))
  {
    $message = "Vous etes deja venu ".$_COOKIE['compteur']." fois
↪<br/>\n";
    $valeur = $_COOKIE['compteur'] + 1;
  }
  else
  {
     $message = "Je vous met un petit cookie<br/>\n";
     $valeur = 1;
  }
setCookie("compteur", $valeur);
echo $message;
```

Exécution :

Cookies en PHP

Mais si on a des informations en plus grand nombre à stocker ou qui revètent un caractère plus sensible, on préférera les stocker essentiellement côté serveur et utiliser le mécanisme plus complet des sessions. Celui-ci consiste à utiliser le tableau associatif _SESSION[] qui permet de stocker toute sorte d'informations associées à ce client (données de type nombres ou chaines, tableaux ou objets PHP).

Considérons une première page mettant en place une session :

```php
<?php
// session1.php
session_start();
if (!isset($_SESSION['cpt']))
    $_SESSION['cpt']=0;
else
    $_SESSION['cpt']++;
echo "Vous avez vu cette page ".$_SESSION['cpt']." fois <br/>\n";
echo "Le SID courant est " . session_id();
echo "<br/> <a href=\"session2.php\">Aller à la page suivante␣
↪session2.php</a>";
```

Puis on va relire les informations stockées en variables de session dans une autre page :

```php
<?php
// session2.php
session_start();
if (!isset($_SESSION['cpt']))
    $_SESSION['cpt']=0;
else
    $_SESSION['cpt']++;
echo "bonjour {$_SESSION['login']} !<br>\n";
```

(suite sur la page suivante)

(suite de la page précédente)

```
echo "vous avez vu cette page " . $_SESSION['cpt'] . " fois<br/>\n";
echo "Votre SID est toujours " . session_id();
echo "<br/> <a href=\"session1.php\">Retour a session1.php</a>";
```

Exécution :

Utilisation variable de session PHP

Champs cachés

Un quatrième mécanisme est employé pour conserver de l'information dans des pages Web elles-mêmes comme l'utilisation de champs cachés : input de type *hidden*. Ces champs peuvent par exemple servir à stocker dans des formulaires HTML un champ spécial dit *csrf token* qui contiendra un identifiant unique temporaire pour se prémunir des attaques de type CSRF : Cross Site Request Forgery dont un exemple est l'envoi d'un mail contenant une image a quelqu'un et cette image est en fait un lien vers une page d'administration sur laquelle le destinataire du mail a des droits particuliers comme editer ou supprimer une ressource.

Cette page pourra déclencher une suppression ou une modification de contenu non souhaitée.

Les principaux Frameworks Web comme Symfony, Laravel en PHP ou Django, Flask en Python prennent en charge la génération automatique de ce token et sa mise en variable de session mais il faut tout de même l'appeler dans les formulaires ou lors de l'utilisation d'Ajax.

3.25 Manipuler XML avec PHP

Le format XML est utilisé de façon très variée. Nous le trouvons dans des services Web, des fichiers de configuration, des formats comme SVG, MathML, docx, odt, etc. Sa manipulation dans un langage comme PHP est donc particulièrement importante.

On peut manipuler XML de différentes manières

- A la main
- Avec XMLWriter/XMLReader
- Avec DOM
- Avec SimpleXML

ou des combinaisons de ces méthodes.

- DOM ou Document Object Model est une méthode qui recrée toute l'arborescence d'un document XML sous forme d'objets PHP. Son utilisation est simple mais elle est couteuse en ressources, en temps d'exécution et un peu verbeuse.
- XMLWriter et XMLReader traitent les fichiers XML a plus bas niveau mais leur utilisation exclusive rend parfois le code délicat à implémenter surtout pour la lecture de fichiers complexes.
- SimpleXML représente une sorte de compromis Simplicité/Performance.
- Traiter des fichiers xml « à la main » est généralement à éviter sauf pour créer des fichiers très simples.

3.25.1 Traitement de fichiers XML à la main :

Observons d'abord comment créer un fichier XML contenant une liste de programmes TV : La lecture de fichiers XML sans API est peu recommandée.

```
<?php
header('Content-Type: text/xml');
print '<?xml version="1.0"?>' . "\n";
print "<programmes>\n";
$programmes = array(
    array('nom'=> 'Simpsons',
        'chaine'=> 'TF18',
        'debut' => '21:00',
        'duree' => '30'),
    array('nom'=> 'Blake et Mortimer',
        'chaine' => 'M54',
        'debut'=>'20:00','duree'=>'60'));
foreach ($programmes as $show) {
    print "\t<show>\n";
    foreach($show as $tag => $data) {
      print "\t<$tag>"
      . htmlspecialchars($data)
      . "\t</$tag>\n";
    }
    print "</show>\n";}
print "</programmes>\n";
```

3.25.2 Exécution :

Ecriture XML à la main

3.25.3 Resultat brut html :

```
<?xml version="1.0"?>
<programmes>
        <show>
        <nom>Simpsons        </nom>
        <chaine>TF18        </chaine>
        <debut>21:00        </debut>
        <duree>30        </duree>
</show>
        <show>
        <nom>Blake et Mortimer        </nom>
        <chaine>M54        </chaine>
        <debut>20:00        </debut>
        <duree>60        </duree>
```

(suite sur la page suivante)

(suite de la page précédente)

```
</show>
</programmes>
```

3.25.4 Ecriture avec XMLWriter :

Un exemple simple pour démarrer :

```
<?php
$xml = new XMLWriter();
$xml->openURI('test.xml');
$xml->startElement('toto');
$xml->writeElement('url', 'http://totototo.com');
$xml->endElement();
$xml->flush();
```

3.25.5 Resultat brut :

```
<toto><url>http://toto.com</url></toto>
```

et si on récupère des données de la table CARNET pour les exporter en XML :

```
<?php
require_once('connexion.php');
$connexion=connect_bd();
$sql="SELECT * from CARNET";
$data=$connexion->query($sql);
$xml= new XMLWriter();
$xml->openUri("contacts.xml");
$xml->startDocument('1.0', 'utf-8');
$xml->startElement('mescontacts');

  while($pers=$data->fetch()){
    $xml->startElement('contact');
    $xml->writeAttribute('id', $pers['ID']);
    $xml->writeElement('prenom',$pers['PRENOM']);
    $xml->writeElement('nom',$pers['NOM']);
    $xml->writeElement('naissance',$pers['NAISSANCE']);
    $xml->endElement();
  }
$xml->endElement();
$xml->endElement();
$xml->flush();
```

3.25.6 Resultat :

```
<?xml version="1.0" encoding="UTF-8"?>
<mescontacts>
    <contact id="1">
        <prenom>JOHN</prenom>
        <nom>SMITH</nom>
        <naissance>1980-12-17</naissance>
    </contact>
    <contact id="2">
        <prenom>JEAN</prenom>
        <nom>DURAND</nom>
        <naissance>1983-01-13</naissance>
    </contact>
    <contact id="3">
        <prenom>JEANNE</prenom>
        <nom>GUDULE</nom>
        <naissance>1967-11-06</naissance>
        </contact>
    <contact id="4">
        <prenom>EMILIO</prenom>
        <nom>ZAPATA</nom>
        <naissance>1956-12-01</naissance>
        </contact>
    <contact id="5">
        <prenom>NICOLAS</prenom>
        <nom>JOURDAIN</nom>
        <naissance>2000-09-10</naissance>
        </contact>
    <contact id="6">
        <prenom>MARIE</prenom>
        <nom>DUPUY</nom>
        <naissance>1986-01-11</naissance>
    </contact>
    <contact id="7">
        <prenom>LOU</prenom>
        <nom>ANDREAS</nom>
        <naissance>1861-02-12</naissance>
    </contact>
    <contact id="9">
        <prenom>Franz</prenom>
        <nom>Kafka</nom>
        <naissance>1883-07-03</naissance>
        </contact>
    <contact id="11">
        <prenom>Joe</prenom>
        <nom>Dalton</nom>
        <naissance>2003-12-06</naissance>
        </contact>
    </mescontacts>
```

3.25.7 Traitements avec DOM :

Ecriture de fichier XML avec DOM en utilisant des données provenant d'une Base de Données. Partons de la table films suivante :

```
CREATE TABLE IF NOT EXISTS `films` (
 `code_film` int(11) NOT NULL AUTO_INCREMENT,
 `titre_original` varchar(50) DEFAULT NULL,
 `titre_francais` varchar(50) DEFAULT NULL,
 `pays` varchar(20) DEFAULT NULL,
 `date` int(11) DEFAULT NULL,
 `duree` int(11) DEFAULT NULL,
 `couleur` varchar(10) DEFAULT NULL,
 `realisateur` int(11) DEFAULT NULL,
 `image` varchar(20) DEFAULT NULL,
 PRIMARY KEY(`code_film`)
)
```

et créons un fichier XML avec les données de cette table en utilisant DOM :

```
<?php
// avec le fichier connexion.php utilisé auparavant
require("connexion.php");
$connexion=connect_bd();
$sql="SELECT * from films limit 10";
$data=$connexion->query($sql);
if ($data){
$document = new DomDocument();
$document->preserveWhiteSpace = false;
$document->formatOutput = true;
// on crée la racine <lesfilms> et on l'insère dans le document
$lesfilms = $document->createElement('lesfilms');
$document->appendChild($lesfilms);

// On boucle  pour tous les films trouvés dans la BD:
while($unfilm=$data->fetch(PDO::FETCH_OBJ))
    {
    $film=$document->createElement('film');
    $film->setAttribute('idreal', $unfilm->realisateur);
    $lesfilms->appendChild($film);
    // on crée l'élément title et on l'ajoute à $film
    $title = $document->createElement('titre');
    $film->appendChild($title);
    // on définit le texte pour $title
    $text=$document->createTextNode(utf8_encode($unfilm->titre_
↪original));
    $title->appendChild($text);

    //crée et ajoute le realisateur a $film
    $realisateur=$document->createElement('date');
```

(suite sur la page suivante)

(suite de la page précédente)

```
    $id=$document->createTextNode($unfilm->date);
    $realisateur->appendChild($id);
    $film->appendChild($realisateur);
  }
  $document->save('myFilms.xml');
  echo "Export XML fini !";
 }

 else { echo "Aucun film dans la base !" ;}
```

3.25.8 Exécution :

Creation XML avec DOM

3.25.9 Resultat :

```
<?xml version="1.0"?>
<lesfilms>
  <film idreal="7">
    <titre>Pandora and the flying Dutchman                    </
↪titre>
    <date>1951</date>
  </film>
  <film idreal="8">
    <titre>Johnny Guitar                                      </
↪titre>
    <date>1954</date>
  </film>
  <film idreal="9">
    <titre>Woman under the Influence (A)                      </
↪titre>
    <date>1974</date>
  </film>
  <film idreal="10">
    <titre>Apartment (The)                                    </
↪titre>
    <date>1960</date>
  </film>
  <film idreal="11">
    <titre>Victor/Victoria                                    </
↪titre>
    <date>1982</date>
  </film>
  <film idreal="12">
    <titre>Modern Times                                       </
↪titre>
```

(suite sur la page suivante)

(suite de la page précédente)

```
    <date>1936</date>
  </film>
  <film idreal="13">
    <titre>M&#xC3;&#xA9;pris (Le)
↪    </titre>
    <date>1963</date>
  </film>
  <film idreal="14">
    <titre>Jour de f&#xC3;&#xAA;te
↪    </titre>
    <date>1948</date>
  </film>
  <film idreal="15">
    <titre>Olvidados (Los)</
↪titre>
    <date>1950</date>
  </film>
  <film idreal="16">
    <titre>West Side Story</
↪titre>
    <date>1961</date>
  </film>
</lesfilms>
```

3.25.10 Relecture avec SimpleXML :

```
<?php

$lesfilms = simplexml_load_file('myFilms.xml');
  foreach ($lesfilms->film as $film) {
    echo "Titre :". utf8_decode($film->titre)."<br/>\n";
    foreach($film->attributes() as $a => $b) {
      echo $a, '="', $b, "\"\n";
    }
    print "Annee : {$film->annee} <br/>\n";
    }
```

3.25.11 Exécution :

Lecture XML avec SimpleXML

3.25.12 Resultat brut :

```
Titre :Pandora and the flying Dutchman
 ↪<br/>
idreal="7"
Annee :  <br/>
Titre :Johnny Guitar
 ↪<br/>
idreal="8"
Annee :  <br/>
Titre :Woman under the Influence (A)
 ↪<br/>
idreal="9"
Annee :  <br/>
Titre :Apartment (The)
 ↪<br/>
idreal="10"
Annee :  <br/>
Titre :Victor/Victoria
 ↪<br/>
idreal="11"
Annee :  <br/>
Titre :Modern Times
 ↪<br/>
idreal="12"
Annee :  <br/>
Titre :Mépris (Le)
 ↪<br/>
idreal="13"
Annee :  <br/>
Titre :Jour de fête
 ↪<br/>
idreal="14"
Annee :  <br/>
Titre :Olvidados (Los)
 ↪<br/>
idreal="15"
Annee :  <br/>
Titre :West Side Story
 ↪<br/>
idreal="16"
Annee :  <br/>
```

3.26 Architecture de type MVC avec PHP

3.26.1 Problème

Lorsqu'un projet augmente, le besoin de s'organiser et de permettre plus de réutilisabilité et de lisibilité demande une certaine méthode. MVC = Modèle Vue Controleur peut être une solution intéressante.

Nous allons commencer à nous familiariser avec les composants d'un Framework MVC et à voir l'utilité de recourir à de tels outils.

Une introduction générale à ce sujet se trouve ici (http ://symfony.com/doc/current/book/from_flat_php_to_symfony2.html)

3.26.2 Du PHP pur aux templates PHP :

Considérons le code suivant en interrogeant la table CARNET vue précemment depuis PHP avec PDO :

```
<?php
  require("connect.php");
  $dsn="mysql:dbname=".BASE.";host=".SERVER;
  try{
  $connexion=new PDO($dsn,USER,PASSWD);
  }
  catch(PDOException $e){
  printf("Echec connexion : %s\n", $e->getMessage());
  exit();
  }
  $sql="SELECT * from CARNET";
  if(!$connexion->query($sql))
  echo "Pb pour acceder au CARNET";
  else
  {
      foreach ($connexion->query($sql) as $row){
          echo $row['NOM']<br/>\n";
      }
  }
?>
```

On peut observer quelques défauts dans le code ci-dessus :

- Réutilisabilté du code très réduite
- Si on fabrique un formulaire avec les entrées du carnet, où doit-on mettre le code correspondant ?

3.26.3 Un template PHP :

On peut améliorer un peu les choses :

```php
<?php
   require("connect.php");
   $dsn="mysql:dbname=".BASE.";host=".SERVER;
   try
   {
       $connexion=new PDO($dsn,USER,PASSWD);
   }
   catch(PDOException $e)
   {
       printf("Echec connexion : %s\n", $e->getMessage());
       exit();
   }
   $sql="SELECT * from CARNET";
   if(!$connexion->query($sql))
       echo "Pb pour acceder au CARNET";
   else
   {
       $amis=Array();
       foreach ($connexion->query($sql) as $row){
           $amis[]=$row;
       }
       require "templates/listeamis.php";
   }
 ?>
```

Avec un template listeamis.php à placer dans templates/listeamis.php :

```php
<!DOCTYPE html>
<html>
<head>
    <title>Liste de mes Amis</title>
</head>
<body>
    <h1>List of friends</h1>
    <ul>
        <?php foreach ($amis as $ami): ?>
        <li>
        <a href="/recherche?nom=<?php echo $ami['ID'] ?>">
        </a>
        </li>
        <?php endforeach; ?>
    </ul>
</body>
</html>
```

On commence ainsi à séparer la présentation du codage « métier ».

3.26.4 Isolons la logique applicative :

```
<?php
  //modele.php
  require("connect.php");

  function connect_db()
  {
      $dsn="mysql:dbname=".BASE.";host=".SERVER;
  try
  {
      $connexion=new PDO($dsn,USER,PASSWD);
  }
  catch(PDOException $e)
  {
  printf("Echec connexion : %s\n",
  $e->getMessage());
  exit();
  }
  return $connexion;
  }
  // Puis
  function get_all_friends()
  {
      $connexion=connect_db();
      $amis=Array();
      $sql="SELECT * from CARNET";
      foreach ($connexion->query($sql) as $row)
      {
          $amis[]=$row;
      }

      return $amis;
  }
  ?>
```

On peut maintenant avoir un **controleur** très simple qui interroge le modèle puis passe les données au template pour affichage.

```
<?php
  //c-list.php
  require_once 'modele.php';

  $amis = get_all_friends();

  require 'templates/listamis.php';
?>
```

3.26.5 Layout :

Il reste une partie non réutilisable dans le code à savoir le layout. Essayons de remédier à ça :

```
<!-- templates/baseLayout.php -->
<!DOCTYPE html>
<html>
<head>
    <title><?php echo $title ?></title>
</head>
<body>
    <?php echo $content ?>
</body>
</html>
```

3.26.6 Héritage de templates :

```
<?php
// templates/t-list.php
  $title = 'Liste des amis';
  ob_start();
?>
  <h1>List de mes amis</h1>
  <ul>
  <?php foreach ($amis as $ami): ?>
  <li>
      <a href="/recherche?nom=<?php echo $ami['nom'] ?>">
          <?php echo $ami['VILLE'] ?>
      </a>
  </li>
  <?php endforeach; ?>
  </ul>
<?php
  $content = ob_get_clean();
  include 'baseLayout.php'
?>
```

Observez l'utilisation de la bufferisation avec *ob_start()* et *ob_get_clean()*. Cette dernière fonction récupère le contenu bufferisé et nettoie ensuite le buffer.

Affichage des détails d'une personne

On va ajouter à notre modèle une fonction pour afficher les détails d'une personne :

```
<?php
  function get_friend_by_id($id)
  {
```

(suite sur la page suivante)

(suite de la page précédente)

```
        $connexion=connect_bd();
        $sql="SELECT * from CARNET where ID=:id";
        $stmt=$connexion->prepare($sql);
        $stmt->bindParam(':id', $id, PDO::PARAM_INT);
        $stmt->execute();
        return $stmt->fetch();
    }
```

On peut maintenant créer un nouveau controleur c-details.php :

```
<?php
  //c-details.php
  require_once 'modele.php';
  $pers = get_friend_by_id($_GET['id']);
  require 'templates/t-details.php';
?>
```

Qui utilise le template :

```
<?php
  //templates/t-details.php
  $title = $pers['NOM'];
  ob_start();
?>
<h1>details sur
<?php echo $pers['PRENOM'].' '.$pers['NOM'] ?>
</h1>
<p>
<?php
  echo ' Ne le '.$pers['NAISSANCE'];
  echo '<br/>Ville:'.$pers['VILLE'];
  $content = ob_get_clean();
  include 'baseLayout.php'
?>
```

Vous pouvez tester en entrant l'URL de c-details.php avec un paramètre id. Le code est similaire à celui du premier template et nous pouvons réutiliser le template de base, mais il subsiste plusieurs problèmes :

— Si le paramètre id n'est pas fourni, notre application va provoquer une erreur.
— Nous n'avons pas de controleur principal.

Regroupons d'abord le code des 2 contrôleurs (c-list.php et c-details.php) dans un fichier unique controllers.php

```
<?php
// controllers.php
  function list_action()
  {
    $amis = get_all_friends();
    require 'templates/t-list.php';
```

(suite sur la page suivante)

(suite de la page précédente)

```
  }

  function detail_action($id)
  {
   $pers = get_friend_by_id($id);
   require 'templates/t-detail.php';
  }
?>
```

Nous pouvons enfin proposer un controleur principal (Front Controller) `index.php` :

```
<?php
  // index.php
  // On charge les modeles et les controleurs
  require_once 'modele.php';
  require_once 'controllers.php';
  // gestion des routes
  $uri = parse_url($_SERVER['REQUEST_URI'], PHP_URL_PATH);
  if ('/index.php' == $uri)
    {
        list_action();
    }
  elseif ('/index.php/detail' == $uri && isset($_GET['id']))
    {
        detail_action($_GET['id']);
    }
    else
    {
        header('Status: 404 Not Found');
        echo '<html><body><h1>Page Not Found</h1></body></html>';
    }
  ?>
```

Nous avons maintenant une structure de ce type :

```
├── connect.php
├── connexion.php
├── controlleur.php
├── modele.php
├── recherche.php
└── templates
    ├── layout.php
    └── listeamis.php
```

On peut améliorer tout cela en intégrant dans un même Objet tout le modèle :

Les templates PHP présentés ne sont pas très simples à utiliser, aussi nous allons étudier un système de templating plus puissant : Twig.

3.27 Templates Twig en PHP

L'installation de Twig se fait grâce à l'outil *composer*

3.27.1 composer

Si composer n'est pas présent dans votre environnement, il faut procéder à son installation. *Composer* servira à beaucoup de tâches courantes dans un projet PHP. On va d'abord créer un répertoire `bin` à la racine de notre `HOME`, placez-y l'exécutable `composer.phar` et renommez le en `composer` (ou faites un lien symbolique).

```
cd
mkdir bin
cd bin
curl -s https://getcomposer.org/installer | php
mv composer.phar composer
```

Si vous êtes derrière un proxy, vérifiez la définition de vos variables d'environnement `http_proxy` et `https_proxy` dans votre `.bashrc` Ajoutez la ligne suivante à votre `.bashrc` pour adapter votre *PATH* :

```
export PATH=$PATH:~/bin
```

de manière à ce que tous les programmes installés dans le répertoire *bin* de votre HOME soient accessibles de n'importe où.

3.27.2 Installation

Installons Twig :

```
composer require twig/twig
```

ou simplement :

```
composer require twig
```

Ceci créera dans le répertoire courant un dossier **vendor** contenant les librairies demandées et les dépendances.

On définit d'abord un template de base, `BaseTemplate.html` :

```
<!DOCTYPE html>
<html lang="fr">
<head>
    {% block head %}
    <meta charset="utf-8">
    <link rel="stylesheet" href="style.css" />
```

(suite sur la page suivante)

(suite de la page précédente)

```
    <title>{% block title %}{% endblock %}</title>
    {% endblock %}
</head>
<body>
<section id="content">
{% block content %}{% endblock %}
</section>
<footer id="footer">
    {% block footer %}
    &copy; Copyright 2020 <a href="http://monsite.com">
    Mon super Site</a>.
    {% endblock %}
</footer>
</body>
</html>
```

Puis un template plus spécialisé qui en hérite, `menu.html` :

```
{% extends "BaseTemplate.html" %}
{% block title %}Menu de la semaine{% endblock %}
{% block head %}
{{ parent() }}
<style>
    .important { color: #336699; }
</style>
{% endblock %}
{% block content %}
    <h1>Menu</h1>
    <p class="important">
        Voici votre menu de la semaine:
        <dl>
            <dt>Lundi</dt>
            <dd>{{Lundi}}</dd>
            <dt>Mardi</dt>
            <dd>{{Mardi}}</dd>
            <dt>Mercredi</dt>
            <dd>{{Mercredi}}</dd>
            <dt>Jeudi</dt>
            <dd>{{Jeudi}}</dd>
        </dl>
    </p>
{% endblock %}
```

Enfin, on utilise ce template dans un fichier `menu.php` en chargeant d'abord l'**autoloader** :

```
<?php
// menu.php
// inclure  l'autoloader
include 'vendor/autoload.php';
```

(suite sur la page suivante)

(suite de la page précédente)

```
try {
    // le dossier ou on trouve les templates
    $loader = new Twig\Loader\FilesystemLoader('templates');

    // initialiser l'environement Twig
    $twig = new Twig\Environment($loader);

    // load template
    $template = $twig->load('Menu.html');

    // set template variables
    // render template
    echo $template->render(array(
        'lundi' => 'Steak Frites',
        'mardi' => 'Raviolis',
        'mercredi' => 'Pot au Feu',
        'jeudi' => 'Couscous',
        'vendredi' => 'Poisson',
    ));

} catch (Exception $e) {
    die ('ERROR: ' . $e->getMessage());
}
```

3.27.3 Affichage des personnes du Carnet

Affichons à présent les personnes du Carnet à l'aide d'un template Twig. On réutilise le fichier modele.php vu précédemment :

```
<?php
// modele.php

class Carnet {
    private static $connexion;
    function __construct(){
        $dsn="mysql:dbname=".BASE.";host=".SERVER;
        try{
                self::$connexion=new PDO($dsn,USER,PASSWD);
            }
        catch(PDOException $e){
            printf("Échec de la connexion : %s\n", $e->
↪getMessage());
            $this->connexion = NULL;
        }
    }
    /** Récupére la liste des contacts sous forme d'un tableau */
```

(suite sur la page suivante)

(suite de la page précédente)

```
    function get_all_friends(){
        $sql="SELECT * from CARNET";
        $data=self::$connexion->query($sql);
        return $data;
    }
    /** Ajoute un contact à la table CARNET */
    function add_friend($data){
        $sql = "INSERT INTO CARNET(NOM,PRENOM,NAISSANCE,VILLE)␣
↪values (?,?,?,?)";
        $stmt = self::$connexion->prepare($sql);
        return $stmt->execute(array($data['nom'],
            $data['prenom'], $data['naissance'],$data['ville']));
    }
    /** Récupére un contact à partir de son ID */
    function get_friend_by_id($id)
    {
        $sql="SELECT * from CARNET where ID=:id";
        $stmt=self::$connexion->prepare($sql);
        $stmt->bindParam(':id', $id, PDO::PARAM_INT);
        $stmt->execute();
        return $stmt->fetch(PDO::FETCH_OBJ);
    }
```

```
<?php
// fichier carnet.php

include 'vendor/autoload.php';

// on inclus le modele
include 'modele.php';
// On instancie un Carnet
$car = new Carnet();
try {
    // le dossier ou on trouve les templates
    $loader = new Twig\Loader\FilesystemLoader('templates');

    // initialiser l'environement Twig
    $twig = new Twig\Environment($loader);

    // load template
    $template = $twig->load('carnet.html');

    // on va instancier le modele
    // et préparer les variables
    // qu'on va passer au template
    require_once("modele.php");
    $contacts = new Contacts();
    $amis = $contacts->get_all_friends();
    $titre = "Liste des contacts";
```

(suite sur la page suivante)

(suite de la page précédente)

```
    // render template
    echo $template->render(array(
        'titre' => $titre,
        'amis' => $amis,
    ));

} catch (Exception $e) {
    die ('ERROR: ' . $e->getMessage());
}
```

et un template carnet.html :

```
{% extends "BaseTemplate.html" %}
{% block title %}Personnes du Carnet {% endblock %}
{% block head %}
{{ parent() }}
<style type="text/css">
    .important { color: #336699; }
</style>
{% endblock %}

{% block content %}

<p align="center" class="Style1">{{titre}}</p>
<table border="2" align="center" cellspacing="0" cellpadding="2" >
<tr bgcolor="#CA9999">
    <td width="50"><strong>numero</strong></td>
    <td width="50"><strong>Nom</strong></td>
    <td width="50"><strong>Prenom</strong></td>
    <td width="30"><strong>Age</strong></td>
    <td width="50"><strong>Ville</strong></td>
</tr>
{% set i=0 %}
{% for ami in amis %}
{% set i=i+1 %}
{% if i is odd %}
<tr bgcolor="#F0F0F0">
{% else %}
    <tr bgcolor="#A6A6A6">
{% endif %}
    <td>{{ami.ID}}</td>
    <td>{{ami.NOM}}</td>
    <td>{{ami.PRENOM}}</td>
    <td>{{ami.NAISSANCE}}</td>
    <td>{{ami.VILLE}}</td>
</tr>
{% endfor %}
</table>
```

(suite sur la page suivante)

(suite de la page précédente)

```
{% endblock %}
```

Ce template est un bien maladroit mais il montre l'expressivité du langage de template Twig avec des boucles, des conditionnelles, des déclarations et calculs avec des variables, etc.

Nous pouvons bien sûr en proposer un plus simple avec le CSS adéquat.

3.27.4 Affichage des Personnes avec un template plus simple

(carnet2.html)

```
{% extends "BaseTemplate.html" %}
{% block title %}Personnes du Carnet{% endblock %}
{% block head %}
{{ parent() }}
<link rel="stylesheet" href="static/css/tabstyle.css" />
{% endblock %}

{% block content %}

<h2>{{titre}}</h2>
<table id="jolie" class="centre" >
<tr>
    <td>numero</td>
    <td>Nom</td>
    <td>Prenom</td>
    <td>Age</td>
    <td>Ville</td>
</tr>
{% for ami in amis %}
    <tr>
    <td>{{ami.ID}}</td>
    <td>{{ami.NOM}}</td>
    <td>{{ami.PRENOM}}</td>
    <td>{{ami.NAISSANCE}}</td>
    <td>{{ami.VILLE}}</td>
</tr>
{% endfor %}
</table>
{% endblock %}
```

avec le style CSS qui va bien sur les tableaux . . .

Nous pouvons ainsi compléter le développement MVC effectué précédemment en utilisant des templates Twig. Voir sur github/roza/php-basic-mvc (https ://github.com/roza/php-basic-mvc/) pour un code plus complet. Mais le système de routage employé est encore très rudimentaire.

Nous pouvons ensuite utliser les outils de microframeworks Web comme Silex, Phalcon ou Slim ou directement ceux du Framework de référence : Symfony pour aller plus loin.

3.28 Le microframework Silex

Nous allons à présent utiliser le microframework PHP Silex dont le système de routage est très simple. Le développement a été arrêté depuis la version 4 du Framework Symfony qui comporte plus de modularité avec *flex* et qui peut notamment s'utiliser comme microframework, comme nous le verrons ensuite.

La documentation de Silex se trouve sur le site Silex (http ://silex.sensiolabs.org/documentation)

3.28.1 Installation et configuration Silex

Vous pouvez installer Silex avec :

```
composer require silex/silex
```

Vous aurez un avertissement indiquant que le développement de Silex a été stoppé. Il n'est pas nécessaire de faire ce tuto, vous pouvez juste regarder comment était fait le routage de Silex et parcourir les exemples rapidement.

Indication : composer permet d'installer des centaines de packages librement disponibles. On les trouve sur Packagist (https ://packagist.org/) .

Si on utilise le serveur Web Apache, ajouter le fichier `.htaccess` suivant au répertoire silex qui va contenir notre app :

```
FallbackResource ~/login/silex/index.php
RewriteBase /~login/silex
```

3.28.2 Routage avec Silex

Installez le fichier index.php dans le dossier silex pour assurer un routage simple.

```
<?php
    require_once __DIR__.'/vendor/autoload.php';
    $app = new Silex\Application();
    $app['debug']=true;
    $app->get('/', function (){
        return 'Bonjour le Monde !';
    });

    $app->get('/hello/{name}', function ($name) use ($app) {
        return 'Hello '.$app->escape($name);
    });

    $app->run();
```

Comme vous pouvez le constater, le routage avec Silex est assez simple et agréable à utiliser. Ajoutons une route contact qui va renvoyer la liste de tous les amis.

```
<?php
 $amis = array(
    1 => array(
        'NAISSANCE'       => '2000-03-29',
        'NOM'    => 'DOE',
        'PRENOM'      => 'CALVIN',
        'VILLE'       => 'Honolulu',
    ),
    2 => array(
        'NAISSANCE'       => '1992-05-27',
        'NOM'    => 'Yoyo',
        'PRENOM'      => 'Albert',
        'VILLE'       => 'Orleans',
    ),
  );

 $app->get('/contact', function () use ($amis) {
    $content = '';
    foreach ($amis as $ami) {
        $content .= $ami['PRENOM'].'  ';
        $content .= $ami['NOM'].' ';
        $content .= $ami['VILLE'].' ';
        $content .= '<br />';
    }

   return $content;
 });
```

ou l'equivalent avec interrogation de la base de données.

Pour une route avec paramètre

```
<?php

 $app->get('/contact/{id}',
 function (Silex\Application $app, $id) use ($amis) {
    if (!isset($amis[$id])) {
        $app->abort(404, "Le contact $id n'existe pas !");
    }

    $ami = $amis[$id];

    return  "<h1>{$ami['NOM']}</h1>".
            "<p>{$ami['VILLE']}</p>".
            "<p>{$ami['NAISSANCE']}</p>";
 });
```

Et on peut également traiter les requêtes POST, PUT, DELETE avec Silex.

CARNET avec Silex

Reprenons notre petit MVC pour en assurer le routage avec Silex. Assurons d'abord la récupération des données :

```
<?php
//modele.php
 require("connect.php");

 function connect_db(){
    $dsn="mysql:dbname=".BASE.";host=".SERVER;
        try{
        $connexion=new PDO($dsn,USER,PASSWD,
            array(PDO::ATTR_PERSISTENT =>true)
            );
        }
        catch(PDOException $e){
        printf("Échec de la connexion : %s\n", $e->getMessage());
        exit();
        }
    return $connexion;
    }

 function get_all_friends(){
    $connexion=connect_db();
    $amis=Array();
    $sql="SELECT * from CARNET";
    $data=$connexion->query($sql);
    while($pers=$data->fetch(PDO::FETCH_ASSOC)){
        $amis[] = $pers;
    }
    return $amis;
 }
```

Puis construisons un fichier index.php qui fera office de contrôleur principal :

Contrôleur principal

```
<?php
// index.php
require_once __DIR__.'/vendor/autoload.php';
require_once 'modele.php';

$app = new Silex\Application();
$app['debug']=true;
```

(suite sur la page suivante)

(suite de la page précédente)

```
$app->get('/contact', function () {
   $content ='<ul>';
   $amis=get_all_friends();
   foreach ($amis as $ami){
       $content.='<li>'.$ami['NOM'].'</li>';
   }
   $content.='</ul>';
   return $content;
});


$app->get('/api/contact', function () {
   $amis=get_all_friends();
   return json_encode($amis);
});
```

Nous proposons ainsi 2 routes :

— /contact qui est destinée à être vue sur un navigateur

— /api/contact qui constitue le début de la mise en place d'un service REST

Nous allons maitenant proposer une troisième route qui renverra tous les contacts en XML.

Service XML

```
<?php
    require_once __DIR__.'/vendor/autoload.php';
    require "modele.php";

    use Symfony\Component\HttpFoundation\Response;
    $app = new Silex\Application();
    $app['debug'] = true;
    $app->get('/contactXML', function(Request $request) use($app,
↪$carnet) {
    $amis = get_all_friends();
    if (!$amis) $app->abort(404, "Contacts inexistants");
    else {
        $xml = new XMLWriter();
        $xml->openMemory();
        $xml->startElement('mescontacts');
        foreach ($amis as $pers){
            $xml->startElement('contact');
            $xml->writeAttribute('id', $pers['ID']);
            $xml->writeElement('prenom',$pers['PRENOM']);
            $xml->writeElement('nom',$pers['NOM']);
            $xml->writeElement('naissance',$pers['NAISSANCE']);
            $xml->endElement();
        }
        $xml->endElement();
        return new Response(
```

(suite sur la page suivante)

(suite de la page précédente)

```
            $xml->outputMemory(),
            200,
            ['Content-Type' => 'text/xml']
            );
        }
    });

$app->run();
```

3.29 Tester une application PHP - TDD

Nous allons à présent nous attaquer à une problématique fondamentale dans toute application qu'elle soit Web, mobile ou autres : Les tests.

3.29.1 TDD

TDD veut dire *Test Driven Development* c'est à dire *Développement dirigé par les tests* C'est une démarche mise en avant en *Méthodologie Agile* Elle consiste en général en l'application des points suivants :

- écrire un test
- vérifier qu'il échoue (car le code qu'il teste n'existe pas)
- écrire juste le code suffisant pour passer le test
- vérifier que le test passe
- procéder à un refactoring du code, c'est-à-dire l'améliorer en gardant les mêmes fonctionnalités.

3.29.2 Intérêt de la démarche :

Les avantages principaux de cette démarche sont :

- Préciser au mieux les spécifications du code et l'API envisagée
- Ceci oblige à faire des choix de conception qui restent parfois trop dans le flou au début du développement
- Plus tard, disposer d'une large base de tests est une riche pour une application car elle permet de vérifier à tout moment que les tests installés ne sont pas mis en défaut par de nouveaux développements ou des refactoring de code

Tous les langages de programmation disposent de Frameworks de tests. Par exemple Java offre JUnit.

PHP quand a lui propose PHPUnit. On peut l'installer via composer :

```
{
"require": {
    "phpunit/phpunit": "6.3.*",
```

(suite sur la page suivante)

(suite de la page précédente)

```
},
"autoload": {
    "psr-0": {
        "Exemple": "src"
        }
    }
}
```

```
composer.phar install
```

Ecrivons à présent notre premier test dans le dossier Tests :

```
<?php
use Exemple\FileLoader;

class FileLoaderTest extends PHPUnit_Framework_TestCase
{
    public function testFileLoaderClassCanBeCreated()
    {
        $f = new FileLoader;
    }
}
```

Pour tester : Placer un fichier *phpunit.xml* à la racine de votre projet contenant :

```
<?xml version="1.0" encoding="UTF-8"?>

<!--

PHPUnit Configuration
=====================
Fichier de configuration standard de phpunit
-->
<phpunit backupGlobals="false"
         backupStaticAttributes="false"
         colors="true"
         bootstrap="vendor/autoload.php"
         convertErrorsToExceptions="true"
         convertNoticesToExceptions="true"
         convertWarningsToExceptions="true"
         processIsolation="false"
         stopOnFailure="false"
         syntaxCheck="false"
>
    <testsuites>
        <testsuite>
            <directory>tests</directory>
        </testsuite>
    </testsuites>
```

(suite sur la page suivante)

(suite de la page précédente)

```
</phpunit>
```

Nous sommes prêts à lancer notre premier test :

```
phpunit
```

Ecrivons maintenant un peu de code pour nous permettre de passer notre premier test :

Nous allons compléter notre test par une vérification qu'un fichier situé dans les textit{fixtures} de test a bien été chargé :

```
<?php
class FileLoaderTest extends PHPUnit_Framework_TestCase
{
    public function testFileLoaderClassCanBeCreated()
    {
        $f = new FileLoader;
    }

    /**
    * Nous voulons récupérer le contenu d'un fichier via
    * une méthode get()
    */
    public function testFileLoaderCanLoadFileContent()
    {
        $f = new FileLoader;
        $r = $f->get(__DIR__.'/fixtures/simple.md');
        $this->assertEquals("Foo\n", $r);
    }
}
```

Puis si nous avons besoin de *Mock Objects*, nous installerons la librairie *mockery* :

```
composer require --dev mockery/mockery
```

3.30 Mettre en place un Web Service REST

3.30.1 Problème

Dans une architecture REST classique, un serveur présente les données d'une table et un Client riche (ou RIA) en JavaScript ou un client Mobile permet de les récupérer et des les afficher. REST signifie *Representational State Transfer*.

Cette architecture permet de réaliser des applications de type *onepage* en reportant sur le client une bonne partie de la logique métier et en offrant des point d'entrée aux clients pour lire des données sur le serveur ou lui en envoyer.

Ces données pourront être envoyées en XML ou de plus en plus aujourd'hui en JSON : JavaScript Object Notation, c'est à dire des objets directement utilisables en JS.

On pose les définitions suivantes :

— RIA = Rich Internet Application
— REST = Representational State Transform
— Logique métier déportée vers le client
— Tâche principale du serveur : Offrir des services de récupération et de stockage de données

Un flux de news pourra ainsi offrir par exemple une ressource du type : /api/v1/news/314159 qui permettra aux clients de récupérer la news numéro 314159 en JSON ou en XML en employant la méthode HTTP GET dans la version 1 de notre API. Dans cet exemple, la news est ici la ressource ou *élément* manipulée dans l'API version 1. La méthode GET sera employée pour récupérer des éléments *individuellement* ou par *Collections*.

La méthode POST sera quand à elle employée pour envoyer vers le serveur un ou plusieurs éléments. D'autres méthodes HTTP pour créer ou modifier complètement (PUT) ou partiellement (PATCH) des éléments ou les effacer (DELETE) seront souvent également disponibles dans l'API.

Les technologies concurrentes à REST sont XML-RPC et SOAP (Microsoft) REST est une façon moderne de concevoir ce genre de service et possède les avantages suivants :

— Bonne montée en charge du serveur
— Simplicité des serveurs (retour aux sources du protocole HTTP)
— Equilibrage de charge
— le serveur offre une API
— les services sont représentés par des URL's donc simplicité et bonne gestion du cache
— Possibilité de décomposer des services complexes en de multiples services plus simples qui communiquent entre eux

Les principes de REST ont été théorisés par Roy Fielding dans sa thèse (http ://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm) :

1. Séparation claire entre Client et Serveur
2. Le client contient la logique métier, le serveur est sans Etat
3. Les réponses du serveur peuvent ou non être mises en cache
4. L'interface doit être simple, bien définie, standardisée
5. Le système peut avoir plusieurs couches comme des proxys, systèmes de cache, etc
6. Eventuellement, les clients peuvent télecharger du code du serveur qui s'exécutera dans le contexte du client

Pour mémoire, une API REST peut offrir les méthodes suivantes :

Méthodes HTTP et REST :

Méthode	Rôle	Code retour HTTP
GET URL	Récupération Element	200
GET URL	Récupération Collection	201
POST URL	Envoi d'Elements	201
DELETE URL	Effacer Element(s)	200
PUT URL	Modifier un Element	200
PATCH URL	Modif. partielle d'Elt.	200

Mais on peut aussi avoir des erreurs comme :

Code Erreur	Description	Signification
400	Bad Request	requête mal formée
404	Not Found	Resource demandée inexistante
401	Unauthorized	Authentification necessaire pour accéder à la resource.
405	Method Not Allowed	Méthode interdite pour cette resource.
409	Conflict	Par exemple, un PUT qui crée une ressource 2 fois
500	Internal Server Error	Autres erreurs du serveur.

Par ailleurs, le serveur REST ne maintient pas d'état, les requêtes sont indépendantes les unes des autres. C'est un retour aux fondamentaux du protocole HTTP qui n'est pas doté de beaucoup de capacités de mémorisation ...

La logique et l'ergonomie de l'application sont gérées côté client. C'est une méthode aujourd'hui plebiscitée pour faire dialoguer des clients (mobiles ou Web) avec des serveurs.

3.31 Exemple de service REST avec PHP

3.31.1 Problème

Nous allons réaliser en PHP l'implémentation d'un service REST qui exposera les données de la table de contact appellée CARNET utilisée dans les autres exemples.

Un contact sera ainsi ainsi accessible à une route du type : /api/v1/contact/12 qui permettra aux clients de récupérer le contact en JSON employant la méthode HTTP GET dans la version 1 de notre API. Dans cet exemple, le contact constitue la *ressource* manipulée dans notre API. La méthode GET sera employée pour récupérer des éléments *individuellement* ou par *Collections*.

Méthode	Action réalisée	URI
GET	Récup. tous les liens	/api/v1/
GET	Récuperation un Element	/api/v1/contact/{id}
GET	Récupération Collection	/api/v1/contact
POST	Creation d'Elements	/api/v1/contact
DELETE	Effacer Element	/api/v1/contact/{id}
PUT	Modifier un Element	/api/v1/contact/{id}
PATCH	Modif. partielle d'Elt.	/api/v1/contact/{id}

La route /api/v1/ en GET renverra la liste des URLs des contacts plutôt que la liste de tous les contacts avec tous leurs détails. Ceci permet d'avoir un serveur REST auto-documenté où la récupération d'une première URL permet en suivant d'obtenir la liste des ressources présentes sur le service avec leurs URLs respectives.

On pourra également paginer les réponses pour ne pas manipuler trop de données simultanément.

Pour assurer le routage simplement nous allons continuer avec Silex (http ://silex.sensiolabs.org/)

Nous implémenterons une autre API REST ensuite avec Symfony 4.

Nous pouvons donc modifier le fichier index.php déjà mis en place comme suit :

```
<?php
require_once __DIR__.'/vendor/autoload.php';
require_once 'modele.php';

$app = new Silex\Application();
$app['debug']=true;

$app->get('/contact', function () {
    $content ='<ul>';
    $amis=get_all_friends();
    foreach ($amis as $ami){
        $content.='<li>'.$ami['NOM'].'</li>';
    }
    $content.='</ul>';
    return $content;
});

$app->get('/api/', function () {
    $amis=get_all_friends_links();
return json_encode($amis);
});

$app->get('/api/contact', function () {
    $amis=get_all_friends();
    return json_encode($amis);
});
```

(suite sur la page suivante)

(suite de la page précédente)

```
?>
```

avec une nouvelle méthode dans modele.php :

```
<?php
  function get_all_friends_links()
  {
       $connexion=connect_db();
       $amis=Array();
       $sql="SELECT * from CARNET";
       $data=$connexion->query($sql);
       while($pers=$data->fetch(PDO::FETCH_ASSOC))
         {
              $res=Array();
              $res['NOM'] = $pers['NOM'];
              $res['URL']=$_SERVER["REQUEST_SCHEME"].'://'.
                          $_SERVER['HTTP_HOST'].
                          $_SERVER['CONTEXT_PREFIX'].
                          '/silex/api/contact/'.$pers['ID'];
              $amis[] = $res;
         }
       return $amis;
  }
  ?>
```

Indication : La vue de base de notre API renvoie maintenant la liste des liens de nos contacts et quelqu'un qui s'y connecte pourra découvrir par la d'autres URLs gérées par notre API. Une bonne API REST se doit d'être **autodocumentée** dans la mesure du possible !

Puis assurons le GET sur l'URI /api/contact/id en ajoutant à index.php :

```
<?php
    $app->get('/api/contact/{id}', function($id) use ($app) {
    $ami = get_friend_by_id($id);
    if (!$ami) $app->abort(404, "Contact inexistant");
    else return json_encode($ami,JSON_PRETTY_PRINT);
    });
?>
```

qui marchera si on ajoute la nouvelle méthode get_friend_by_id() au modèle :

```
<?php
    function get_friend_by_id($id)
    {
      $connexion=connect_db();
      $sql="SELECT * from CARNET where ID=:id";
      $stmt=$connexion->prepare($sql);
```

(suite sur la page suivante)

(suite de la page précédente)

```
      $stmt->bindParam(':id', $id, PDO::PARAM_INT);
      $stmt->execute();
      return $stmt->fetch(PDO::FETCH_OBJ);
    }
?>
```

Continuons avec la méthode http DELETE sur la même route en ajoutant à index.php :

```
<?php
    $app->delete('/api/contact/{id}', function($id) use ($app) {
    $ami = get_friend_by_id($id);
    if (!$ami)
        $app->abort(404, "Contact inexistant");
    else {
        delete_friend_by_id($id);
        return json_encode($ami,JSON_PRETTY_PRINT);
    }
    });
?>
```

en ajoutant au modèle :

```
<?php
    function delete_friend_by_id($id)
    {
      $connexion=connect_db();
      $sql="Delete from CARNET where ID=:id";
      $stmt=$connexion->prepare($sql);
      $stmt->bindParam(':id', $id, PDO::PARAM_INT);
      $stmt->execute();
      return $stmt->fetch(PDO::FETCH_OBJ);
    }
?>
```

Enfin le POST doit nous permettre d'envoyer un nouveau contact pour peupler la table CARNET sur la route /api/contact. Nous assurons d'abord la récupération du contenu json sous la forme d'un tableau PHP avec la méthode before de Silex :

```
<?php
        $app->before(function (Request $request) {
            if (0 === strpos($request->headers->get('Content-Type'),
↪ 'application/json'))
            {
                $data = json_decode($request->getContent(), true);
                $request->request->replace(is_array($data) ? $data␣
↪: array());
            }
        });
?>
```

Puis la méthode post proprement dite :

```
<?php
    $app->post('/api/contact', function (Request $request) use (
↪$app) {
      $data = $request->request->all();
      add_friends($data);
      return new Response(json_encode($data), 200, array('Content-
↪Type' => 'application/json'));
    });
?>
```

N'oubliez pas de faire appel aux objets Request et Response au début du fichier index.php :

```
<?php
  use Symfony\Component\HttpFoundation\Request;
  use Symfony\Component\HttpFoundation\Response;
?>
```

Il ne reste plus qu'à ajouter au modèle :

```
<?php
    function add_friends($data)
    {
      $connexion=connect_db();
      $sql="INSERT INTO CARNET(NOM,PRENOM,NAISSANCE,VILLE) values (?
↪,?,?,?)";
      $stmt=$connexion->prepare($sql);
      return $stmt->execute(array($data['NOM'], $data['PRENOM'],
↪$data['NAISSANCE'],$data['VILLE']));
    }
?>
```

Il n'y a plus qu'à implémenter un PUT et surtout à Tester ! !

3.32 Tester une API REST avec votre navigateur ou avec curl

Pour tester notre API nous pouvons dans un premier temps utiliser l'extension **Postman** de Chrome ou l'extension **RESTClient** pour Firefox.

> **Avertissement :** Attention à bien désactiver les proxys dans vos navigateurs si vous voulez utiliser ces extensions. Pour chrome on peut le lancer en ligne de commande avec l'option –no-proxy-server

Si on veut une solution en ligne de commande, *curl* permet de manipuler les différentes méthodes HTTP. La syntaxe n'est pas idéale mais on peut tester toutes les méthodes HTTP.

Avertissement : Si vous avez déclaré des variables d'environnement http_proxy ou https_proxy, il vaut mieux les désactiver pour que curl n'essaie pas de passer par un proxy ce qui serait problématique pour certaines de ces requêtes curl :

```
unset http_proxy
unset https_proxy
```

3.32.1 Pour tester un GET :

```
curl -i http://localhost/silex/api/v1/contact/2
```

ou si on utilise le module user_dir d'Apache.

```
curl -i http://localhost/~login/silex/api/v1/contact/2
```

Avertissement : Attention dans le cas ou on utilise *user_dir*, les réglages pour utiliser toutes les methodes du protocole HTTP/1.1 peuvent s'avérer délicats.

Notamment le fichier /etc/apache2/mods-available/userdir.conf doit être modifié comme suit (les user_dir sont dans www, syntaxe pour Apache > 2.2) :

```
<IfModule mod_userdir.c>
    UserDir www
    UserDir disabled root

    <Directory /home/*/www>
            AllowOverride All
            Options MultiViews Indexes SymLinksIfOwnerMatch
            <Limit GET POST PUT DELETE OPTIONS>
                    Require all granted
            </Limit>
            <LimitExcept GET POST PUT DELETE OPTIONS>
                    Require all denied
            </LimitExcept>
    </Directory>
</IfModule>
```

et il faut aussi dans votre dossier sous voter user_dir (par exemple ~/www/silex) avoir le .htaccess suivant :

```
<Limit GET POST DELETE PUT OPTIONS>
allow from all
</Limit>
```

(suite sur la page suivante)

(suite de la page précédente)

```
FallbackResource /~roza/silex/index.php
RewriteBase /~roza/silex
```

Dans **tous les cas**, préciser si besoin dans votre php.ini ou un fichier équivalent :

```
always_populate_raw_post_data = -1
```

3.32.2 Pour tester un POST :

```
curl -i -H "Content-Type: application/json" -X POST
 -d '{"NOM":"Dalton", "PRENOM":"joe", "NAISSANCE":"2000-08-15",
 ↪"VILLE":"Orleans"}' http://localhost/silex/api/contact
```

3.32.3 Test un PUT :

```
curl -i -H "Content-Type: application/json" -X PUT -d '{"done":true}
 ↪' http://localhost/silex/api/contact/5
```

3.32.4 Test de DELETE :

```
curl -i -H "Content-Type: application/json" -X "DELETE"
http://localhost/silex/api/contact/7
```

3.33 Tester une API

3.33.1 Tester une API avec Postman

C'est le moyen le plus simple pour tester une API. On l'installe en téléchargeant l'App de Postman (https ://www.getpostman.com/apps) pour votre OS. On peut alors se constituer des collections de requêtes HTTP pour tester une API REST spécifique. Il est également possible de rejouer automatiquement une batterie de tests Postman grâce au paquet node newmann (https ://www.npmjs.com/package/newman).

3.33.2 Tester une API REST avec Guzzle

Les tests de notre API REST avec curl sont peu lisibles. Si vous préférez disposer d'une batterie de tests en PHP, vous pouvez utiliser une librairie spécialisée comme Guzzle. On peut installer cette dernière avec composer :

```
composer require guzzlehttp/guzzle
```

3.34 Composer et Symfony

Nous allons à présent nous familiariser avec les outils et composants d'un Framework de référence : Symfony qui est très modulaire et permet d'installer des composants très riches comme SwiftMailer pour envoyer des mails, FOSUserBundle pour gérer des utilisateurs, FOSRESTBundle ou APIPlatform pour réaliser rapidement une API complète. Le Framework Symfony 4 est basé sur un Micro-noyau (Micro-Kernel) 70% plus léger que le noyau de Symfony 3. La version actuelle est la 5.

Une introduction générale à ce framework se trouve ici (https ://symfony.com/doc/current/index.html)

La gestion des dépendances se fait à présent grâce à l'outil Symfony *Flex* qui permet d'établir des recettes ou *recipes* décrivant les dépendances et la configuration d'un projet. L'outil de base est toujours *composer*.

Indication : Composer permet d'installer des centaines de packages librement disponibles. On les trouve sur Packagist (https ://packagist.org/) . Il permet de gérer les dépendances d'un projet et également de créer le squellette d'une application Symfony 4 et également d'installer des *recettes flex* (recipes) comme *composer require mailer* pour installer SwiftMailer ou *composer require api* pour installer APIPlatform avec toutes ses dépendances.

Si on veut juste installer un composant simple comme *HTTPFoundation », on place à la racine du dossier de travail le fichier *composer.json* suivant :

```
{
"require": {
    "symfony/http-foundation": "~4.2"
    }
}
```

Ceci indique que nous n'installons pour l'instant que ce seul composant et que nous demandons la dernière version stable de la branche **4** pour http-foundation. Puis utilisons `composer` pour installer les composants demandés :

```
composer update -o
```

Indication : Notez l'utilisation de l'option -o de composer pour *optimize-autoloader* qui optimise « au mieux » le chargement automatique des classes.

On peut aussi ne pas se fatiguer à écrire le fichier *composer.json* et laisser faire composer tout seul :

```
composer require symfony/http-foundation
```

qui créera de lui-même ce fichier *composer.json* tout en installant le composant demandé dans le dossier *vendor* du répertoire courant.

3.34.1 HttpFoundation :

Les 2 principaux composants de HttpFoundation à savoir `Request` et `Response` sont alors prêts à l'emploi.

Remarquez l'usage des espaces de nommages en PHP semblables à ceux du C++ ou aux import de packages en java pour éviter des conflits de nommages entre différents *vendor* c'est à dire différentes entités fournissant du code à votre projet. L'"*autoloader* permet le chargement dynamique des classes concernées lors de leur utilisation.

```php
<?php
  // chargement autoloader
  require_once __DIR__.'/vendor/autoload.php';

  use Symfony\Component\HttpFoundation\Request;
  use Symfony\Component\HttpFoundation\Response;

  // Actual request :
  //$request = Request::createFromGlobals();
  // fake request
  $request = Request::create('/essai.php?name=Zozo');

  // URI demandee (sans les parametres)
  $path=$request->getPathInfo();

 // recup de variables en GET
  $nom = $request->query->get('name','World');
  $prenom = $request->query->get('surname','Joe');
  echo "Bonjour $prenom $nom<br/>";
```

On peut aussi récupérer d'autres informations sur le Client et fabriquer une réponse :

```php
<?php
require_once __DIR__.'/vendor/autoload.php';
use Symfony\Component\HttpFoundation\Request;
use Symfony\Component\HttpFoundation\Response;

// toujours avec
$request = Request::create('/essai.php?name=Zozo');

// recup  variables SERVER
$host=$request->server->get('HTTP_HOST');

// get  COOKIES
```

(suite sur la page suivante)

(suite de la page précédente)

```
$request->cookies->get('PHPSESSID');

// HTTP headers
$headers = $request->headers->get('host');
$content_type = $request->headers->get('content_type');

 // URI demandee (sans les parametres)
 $path = $request->getPathInfo();

$method = $request->getMethod();  //GET, POST, PUT, DELETE etc.
$langs = $request->getLanguages();
$IP = $request->getClientIp();
$response =
new Response($IP." ".$host." ".$path." ".$headers."
".$content_type." ".$method." ".$langs[0]);
$response->send();
```

3.34.2 Créer une application Symfony

Il existe 2 façons de procéder :

— en utilisant l'installeur `Symfony`
— en utilisant composer

voir par exemple la doc de symfony (https ://symfony.com/doc/current/setup.html#creating-symfony-applications) ou cette question stackoverflow (https ://stackoverflow.com/questions/25749655/how-do-i-create-a-project-based-on-a-specific-version-of-symfony-using-composer/27766284#27766284)

L'installeur symfony

Pour créer une application web traditionnelle :

```
symfony new --full my_project
```

Pour créer une app plus légère comme un microservice, une app console ou une API :

```
symfony new my_project
```

Pour installer la version LTS courante :

```
symfony new my_project --version=lts
```

Pour installer la dernière version stable :

```
symfony new my_project --version=stable
```

Pour installer la version de développement :

```
symfony new my_project --version=next
```

et pour installer une version spécifique :

```
symfony new my_project --version=4.4
```

Avec composer :

Pour créer une application web traditionnelle (ici en version 4.4) :

```
composer create-project symfony/website-skeleton my_project ^4.4.0
```

Pour créer une app plus légère comme un microservice, une app console ou une API (version 4.4 ici) :

```
composer create-project symfony/skeleton my_project ^4.4.0
```

3.34.3 Squellette d'application Symfony :

Créons un répertoire de travail mvc-sf dans votre dossier Web, par exemple www ou ~/www (ou ~/public_html) si on utilise user_dir d'Apache ou plus simplement n'importe où si on utilise le serveur Web embarqué de PHP ou celui fourni par Symfony.

et créons la trame d'une application Symfony (sf) de type *microservice* à l'aide de *symfony* :

```
symfony new hello-sf
```

ou à l'aide de *composer* :

```
composer create-project symfony/skeleton hello-sf
```

Qui est beaucoup plus verbeux et vous montre qu'il y a tout de même pas mal de packages installés au passage :

```
Installing symfony/skeleton (v5.0.99)
- Installing symfony/skeleton (v5.0.99): Loading from cache
Created project in hello-sf
Loading composer repositories with package information
Updating dependencies (including require-dev)
Package operations: 1 install, 0 updates, 0 removals
- Installing symfony/flex (v1.6.2): Loading from cache
Symfony operations: 1 recipe (f2f9ab59a41987856b579e7321f4971e)
- Configuring symfony/flex (>=1.0): From github.com/symfony/
 ↪recipes:master
Loading composer repositories with package information
Updating dependencies (including require-dev)
```

(suite sur la page suivante)

(suite de la page précédente)

```
Restricting packages listed in "symfony/symfony" to "5.0.*"
Package operations: 27 installs, 0 updates, 0 removals
- Installing psr/container (1.0.0): Loading from cache
- Installing symfony/service-contracts (v2.0.1): Loading from cache
- Installing symfony/polyfill-php73 (v1.14.0): Loading from cache
- Installing symfony/polyfill-mbstring (v1.14.0): Loading from cache
- Installing symfony/console (v5.0.5): Loading from cache
- Installing symfony/dotenv (v5.0.5): Loading from cache
- Installing symfony/routing (v5.0.5): Loading from cache
- Installing symfony/finder (v5.0.5): Loading from cache
- Installing symfony/filesystem (v5.0.5): Loading from cache
- Installing psr/log (1.1.3): Loading from cache
- Installing symfony/polyfill-intl-idn (v1.14.0): Loading from cache
- Installing symfony/mime (v5.0.5): Loading from cache
- Installing symfony/http-foundation (v5.0.5): Loading from cache
- Installing psr/event-dispatcher (1.0.0): Loading from cache
- Installing symfony/event-dispatcher-contracts (v2.0.1): Loading␣
 ↪from cache
- Installing symfony/event-dispatcher (v5.0.5): Loading from cache
- Installing symfony/var-dumper (v5.0.5): Loading from cache
- Installing symfony/error-handler (v5.0.5): Loading from cache
- Installing symfony/http-kernel (v5.0.5): Loading from cache
- Installing symfony/dependency-injection (v5.0.5): Loading from␣
 ↪cache
- Installing symfony/config (v5.0.5): Loading from cache
- Installing symfony/var-exporter (v5.0.5): Loading from cache
- Installing psr/cache (1.0.1): Loading from cache
- Installing symfony/cache-contracts (v2.0.1): Loading from cache
- Installing symfony/cache (v5.0.5): Loading from cache
- Installing symfony/framework-bundle (v5.0.5): Loading from cache
- Installing symfony/yaml (v5.0.5): Loading from cache
Writing lock file
Generating autoload files
Symfony operations: 3 recipes (f2f9ab59a41987856b579e7321f4971e)
- Configuring symfony/framework-bundle (>=4.4): From github.com/
 ↪symfony/recipes:master
- Configuring symfony/console (>=4.4): From github.com/symfony/
 ↪recipes:master
- Configuring symfony/routing (>=4.2): From github.com/symfony/
 ↪recipes:master
Executing script cache:clear [OK]
Executing script assets:install public [OK]
```

On peut aussi créer une application Web plus complète (soyez un peu plus patients dans ce cas . . .) avec :

```
composer create-project symfony/website-skeleton sf-full-project
```

ou

```
symfony new --full sf-full-project
```

Qui installera beaucoup plus de paquets, y compris l'ORM *doctrine*.

Veillez à bien avoir une version à jour de composer et si besoin :

```
composer self-update
```

La structure du projet *skeleton* hello-sf ainsi créé est la suivante :

```
hello-sf/
├── bin
│   └── console
├── composer.json
├── composer.lock
├── config
├── ├── boostrap.php
│   ├── bundles.php
│   ├── packages
│   ├── routes
│   ├── routes.yaml
│   └── services.yaml
├── public
│   └── index.php
├── src
│   ├── Controller
│   └── Kernel.php
├── symfony.lock
├── var
│   ├── cache
│   └── log
└── vendor
    ├── autoload.php
    ├── bin
    ├── composer
    ├── psr
    └── symfony
```

Le répertoire *bin* contient l'outil *console* qui permet d'effectuer les tâches de routine pour créer ou gérer un projet. Le répertoire *config* contient les fichiers de configuration. Le répertoire *public* contient le fichier index de l'application Le dossie *src* les controleurs, le Kernel mais aussi les entités etc. Le dossier *var* contient les cache et les logs et le dossier *vendor* les classes des Bundles installés comme http-foundation.

Vous pouvez consulter le fichier *symfony.lock* qui se trouve à la racine du dossier hello-sf pour voir la liste complète des dépendances installées :

```
{
    "php": {
        "version": "7.3"
    },
```

(suite sur la page suivante)

(suite de la page précédente)

```
    "psr/cache": {
        "version": "1.0.1"
    },
    "psr/container": {
        "version": "1.0.0"
    },
    "psr/event-dispatcher": {
        "version": "1.0.0"
    },
    "psr/log": {
        "version": "1.1.3"
    },
    "symfony/cache": {
        "version": "v5.0.5"
    },
    "symfony/cache-contracts": {
        "version": "v2.0.1"
    },
    "symfony/config": {
        "version": "v5.0.5"
    },
    "symfony/console": {
        "version": "4.4",
        "recipe": {
            "repo": "github.com/symfony/recipes",
            "branch": "master",
            "version": "4.4",
            "ref": "ea8c0eda34fda57e7d5cd8cbd889e2a387e3472c"
        },
        "files": [
            "bin/console",
            "config/bootstrap.php"
        ]
    },
    "symfony/dependency-injection": {
        "version": "v5.0.5"
    },
    "symfony/dotenv": {
        "version": "v5.0.5"
    },
    "symfony/error-handler": {
        "version": "v5.0.5"
    },
    "symfony/event-dispatcher": {
        "version": "v5.0.5"
    },
    "symfony/event-dispatcher-contracts": {
        "version": "v2.0.1"
    },
```

(suite sur la page suivante)

(suite de la page précédente)


```
    "symfony/filesystem": {
        "version": "v5.0.5"
    },
    "symfony/finder": {
        "version": "v5.0.5"
    },
    "symfony/flex": {
        "version": "1.0",
        "recipe": {
            "repo": "github.com/symfony/recipes",
            "branch": "master",
            "version": "1.0",
            "ref": "c0eeb50665f0f77226616b6038a9b06c03752d8e"
        },
        "files": [
            ".env"
        ]
    },
    "symfony/framework-bundle": {
        "version": "4.4",
        "recipe": {
            "repo": "github.com/symfony/recipes",
            "branch": "master",
            "version": "4.4",
            "ref": "23ecaccc551fe2f74baf613811ae529eb07762fa"
        },
        "files": [
            "config/bootstrap.php",
            "config/packages/cache.yaml",
            "config/packages/framework.yaml",
            "config/packages/test/framework.yaml",
            "config/routes/dev/framework.yaml",
            "config/services.yaml",
            "public/index.php",
            "src/Controller/.gitignore",
            "src/Kernel.php"
        ]
    },
    "symfony/http-foundation": {
        "version": "v5.0.5"
    },
    "symfony/http-kernel": {
        "version": "v5.0.5"
    },
    "symfony/mime": {
        "version": "v5.0.5"
    },
    "symfony/polyfill-intl-idn": {
        "version": "v1.14.0"
```


(suite sur la page suivante)

(suite de la page précédente)

```
    },
    "symfony/polyfill-mbstring": {
        "version": "v1.14.0"
    },
    "symfony/polyfill-php73": {
        "version": "v1.14.0"
    },
    "symfony/routing": {
        "version": "4.2",
        "recipe": {
            "repo": "github.com/symfony/recipes",
            "branch": "master",
            "version": "4.2",
            "ref": "683dcb08707ba8d41b7e34adb0344bfd68d248a7"
        },
        "files": [
            "config/packages/prod/routing.yaml",
            "config/packages/routing.yaml",
            "config/routes.yaml"
        ]
    },
    "symfony/service-contracts": {
        "version": "v2.0.1"
    },
    "symfony/var-dumper": {
        "version": "v5.0.5"
    },
    "symfony/var-exporter": {
        "version": "v5.0.5"
    },
    "symfony/yaml": {
        "version": "v5.0.5"
    }
}
```

3.34.4 Squellette d'application web Symfony

Si on crée une application complète avec :

```
symfony new --full my_project
```

On obtient une sctructure plus complète :

```
my_project
|-- bin
|   |-- console
|   `-- phpunit
|-- composer.json
```

(suite sur la page suivante)

(suite de la page précédente)

```
|-- composer.lock
|-- config
|   |-- bootstrap.php
|   |-- bundles.php
|   |-- packages
|   |-- routes
|   |-- routes.yaml
|   `-- services.yaml
|-- phpunit.xml.dist
|-- public
|   `-- index.php
|-- src
|   |-- Controller
|   |-- Entity
|   |-- Kernel.php
|   |-- Migrations
|   `-- Repository
|-- symfony.lock
|-- templates
|   `-- base.html.twig
|-- tests
|   `-- bootstrap.php
|-- translations
|-- var
|   |-- cache
|   `-- log
`-- vendor
    |-- autoload.php
    |-- bin
    |-- composer
    |-- doctrine
    |-- easycorp
    |-- egulias
    |-- jdorn
    |-- monolog
    |-- nikic
    |-- ocramius
    |-- phpdocumentor
    |-- psr
    |-- sensio
    |-- symfony
    |-- twig
    |-- webmozart
    `-- zendframework
```

Nous allons ensuite compléter cette application Symfony par un CRUD puis une API.

3.35 Débuts avec Symfony

3.35.1 Préparatifs

Squellette d'application Symfony

Partons de la trame d'application de type *microservice* déjà créée à l'aide de *composer* :

```
composer create-project symfony/skeleton hello-sf
```

Serveur Web embarqué de Symfony

Lançons le serveur web embarqué de Symfony.

```
symfony server:start -d
```

Ce qui lance le serveur emabrqué de PHP en arrière plan.

Pour le stopper :

```
symfony server:stop
```

Si on veut le lancer au premier plan (ou si l'extension pcntl n'est pas installée), on le lance au premier plan avec :

```
symfony server:start
```

Pour le stopper :

```
Ctrl+C
```

Visitons 127.0.0.1:8000 (https ://127.0.0.1 :8000) (dans mon cas) et constatons que Symfony Marche ! !

3.35.2 Un Controleur et une route simple

routes

Observer le contenu de `routes.yaml` dans le dossier config. Laisser ce code commenté et ajoutez-y le code suivant :

```
hello:
    path: /hello
    controller: App\Controller\HelloController::sayHello
```

contrôleur

Puis placer le contrôleur suivant dans src/Controller/HelloController.php :

```
<?php

namespace App\Controller;

use Symfony\Component\HttpFoundation\Response;

class HelloController
{
    public function sayHello()
    {
        return new Response('Hello!');
    }
}
```

On teste : 127.0.0.1:8000/hello (https ://127.0.0.1 :8000/hello)

3.35.3 Annotations et routes paramétrées

annotations

Au lieu de placer toutes les routes de l'application dans un seul fichier, il peut-être plus souple d'utiliser les *annotations* dans le code même du contrôleur pour plus de commodité :

Commençons par les installer :

```
composer require annotations
```

— Recommentez tout le contenu de *routes.yaml*
— Puis annotez votre contrôleur :

routes paramétrées

Ajoutons une autre route paramétrée dans notre contrôleur :

```
<?php
    namespace App\Controller;

    use Symfony\Component\HttpFoundation\Response;
    use Symfony\Component\Routing\Annotation\Route;

    class HelloController
    {
        /**
        * @Route("/hello")
```

(suite sur la page suivante)

(suite de la page précédente)

```
        */
        public function sayHello()
        {
            return new Response('Hello!');
        }
        /**
        * @Route("/bonjour/{nom}")
        */
        public function bonjour($nom)
        {
            return new Response("Bonjour $nom !");
        }
    }
```

debug des routes

On peut lister toutes ses routes :

```
php bin/console debug:router
```

et obtenir :

Name	Method	Scheme	Host	Path
app_hello_sayhello	ANY	ANY	ANY	/hello
app_hello_bonjour	ANY	ANY	ANY	/bonjour/{nom}

la commande bin/console

Cette commande nous permet d'avoir des informations sur notre projet, de faire des actions primaires dessus et de le débugger. Afin d'obtenir la liste des options offertes par cette commande :

```
php bin/console
```

Prenez le temps de lire la documentation de chaque commande et d'essayer de comprendre ce que chacune d'elle fait.

Remarquez la commande que nous avons utilisée pour lister les routes de notre projet : debug :router Displays current routes for an application

Puis retester : 127.0.0.1:8000/bonjour/toto (https ://127.0.0.1 :8000/bonjour/toto)

3.35.4 Utiliser des templates Twig dans sf

Nous voudrions à présent utiliser des templates Twig dans notre application.

installation

Commençons par utiliser la recette flex pour l'installer :

```
composer require twig
```

contrôleur avec twig

Puis changeons notre contrôleur pour hériter de `AbstractController` :

```
<?php

    namespace App\Controller;

    use Symfony\Component\HttpFoundation\Response;
    use␣
↪Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
    use Symfony\Component\Routing\Annotation\Route;

    class HelloController extends AbstractController
    {
        /**
        * @Route("/hello")
        */
        public function sayHello()
        {
            return new Response('Hello!');
        }
        /**
        * @Route("/bonjour/{nom}")
        */
        public function bonjour($nom)
        {
            //return new Response("Bonjour $nom !");
            return $this->render('bonjour.html.twig', [
                        'nom' => $nom,
                        ]);
        }
    }
```

template twig

et mettons en place le template correspondant `bonjour.html.twig` dans le dossier « templates » :

```
{# templates/bonjour.html.twig #}
{% extends 'base.html.twig' %}
```

(suite sur la page suivante)

(suite de la page précédente)

```
{% block body %}
    <h1>Bonjour {{ nom }}</h1>
{% endblock %}
```

avec un `base.html.twig` du type :

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>{% block title %}Welcome!{% endblock %}</title>
        {% block stylesheets %}{% endblock %}
    </head>
    <body>
        {% block body %}{% endblock %}
        {% block js %}{% endblock %}
    </body>
</html>
```

Retestons : 127.0.0.1:8000/bonjour/toto (https ://127.0.0.1 :8000/bonjour/toto)

Memo twig

```
php bin/console debug:twig
```

Vous donnera la liste des Fonctions, Filtres et Tests disponibles dans les templates Twig.

3.36 Doctrine et Symfony

3.36.1 ORM Doctrine

ORM

ORM signifie Object-Relationnal Mapper. Un ORM sert à offrir une couche d'abstraction de connexion à toutes les BD relationnelles (comme PDO) mais aussi des facilités pour réaliser les requêtes courantes sans descendre au niveau des requêtes SQL et pour générer automatiquement des entités dans le langage utilisé avec les *getters* et *setters* correspondants.

installation

Il suffit de taper :

```
composer req orm
```

pour disposer de l'installation de l'ORM standard de Symfony qui est *Doctrine* Voir sa documentation (https ://www.doctrine-project.org/)

3.36.2 symfony maker-bundle

Il permet de fabriquer des entités, des contrôleurs, des CRUD, des tests etc.

installation

```
composer req maker-bundle --dev
```

sous-commandes disponibles

```
php bin/console list make
```

ce qui donne :

```
Symfony 5.0.7 (env: dev, debug: true)

Usage:
command [options] [arguments]

Options:
-h, --help            Display this help message
-q, --quiet           Do not output any message
-V, --version         Display this application version
    --ansi            Force ANSI output
    --no-ansi         Disable ANSI output
-n, --no-interaction  Do not ask any interactive question
-e, --env=ENV         The Environment name. [default: "dev"]
    --no-debug        Switches off debug mode.
-v|vv|vvv, --verbose  Increase the verbosity of messages: 1 for␣
↪normal output, 2 for more verbose output and 3 for debug

Available commands for the "make" namespace:
make:auth                   Creates a Guard authenticator of␣
↪different flavors
make:command                Creates a new console command class
make:controller             Creates a new controller class
make:crud                   Creates CRUD for Doctrine entity class
make:entity                 Creates or updates a Doctrine entity␣
↪class, and optionally an API Platform resource
make:fixtures               Creates a new class to load Doctrine␣
↪fixtures
make:form                   Creates a new form class
make:functional-test        Creates a new functional test class
```

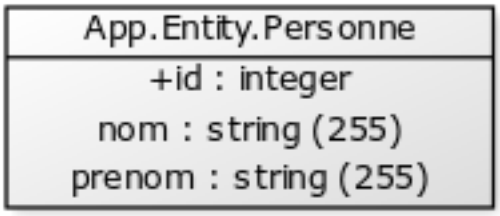
(suite sur la page suivante)

(suite de la page précédente)

```
make:message                  Creates a new message and handler
make:messenger-middleware     Creates a new messenger middleware
make:migration                Creates a new migration based on␣
↪database changes
make:registration-form        Creates a new registration form system
make:reset-password           Create controller, entity, and␣
↪repositories for use with symfonycasts/reset-password-bundle.
make:serializer:encoder       Creates a new serializer encoder class
make:serializer:normalizer    Creates a new serializer normalizer␣
↪class
make:subscriber               Creates a new event subscriber class
make:twig-extension           Creates a new Twig extension class
make:unit-test                Creates a new unit test class
make:user                     Creates a new security user class
make:validator                Creates a new validator and constraint␣
↪class
make:voter                    Creates a new security voter class
```

Entités

Créons une entité `Personne` de ce type :




```
php bin/console make:entity
```

— Répondez aux questions pour ajouter des champs nom et prenom de type string dans `Personne`.
— Vérifiez la création du code correspondant dans src/Entity.

```php
<?php

namespace App\Entity;

use Doctrine\ORM\Mapping as ORM;

/**
* @ORM\Entity(repositoryClass="App\Repository\PersonneRepository")
*/
class Personne
{
    /**
    * @ORM\Id()
```

(suite sur la page suivante)

(suite de la page précédente)

```
     * @ORM\GeneratedValue()
     * @ORM\Column(type="integer")
     */
    private $id;

    /**
     * @ORM\Column(type="string", length=255)
     */
    private $nom;

    /**
     * @ORM\Column(type="string", length=255)
     */
    private $prenom;

    public function getId(): ?int
    {
        return $this->id;
    }

    public function getNom(): ?string
    {
        return $this->nom;
    }

    public function setNom(string $nom): self
    {
        $this->nom = $nom;

        return $this;
    }

    public function getPrenom(): ?string
    {
        return $this->prenom;
    }

    public function setPrenom(string $prenom): self
    {
        $this->prenom = $prenom;

        return $this;
    }
}
```

Config

Réglons la configuration de la base dans .env sur SQLite :

```
###> doctrine/doctrine-bundle ###
DATABASE_URL=sqlite:///%kernel.project_dir%/var/carnet.db?
 ↪charset=utf8mb4
###< doctrine/doctrine-bundle ###
```

Interaction avec la BD

Initialisons la base :

```
./bin/console doctrine:database:create
./bin/console doctrine:schema:update --force
```

easy_admin

Pour faciliter l'interaction avec la BD, installons easy_admin :

```
composer req admin
```

Puis visitez : localhost:8000/admin (http ://localhost :8000/admin)

Nous obtenons une erreur, complétons donc config/packages/easy_admin.yaml :

```
easy_admin:
    entities:
    # List the entity class name you want to manage
        - App\Entity\Personne
```

et vous pourrez ensuite ajouter facilement des personnes dans l'admin !

générateur de CRUD

Nous aurions pu aussi utiliser le générateur de CRUD :

```
php bin/console make:crud
```

en choisissant l'entité `Personne`.

Cela fabrique automatiquement un CRUD avec le contrôleur et les templates nécessaires. Intéressant pour inspecter le code généré même si visuellement easy_admin est plus abouti . . .

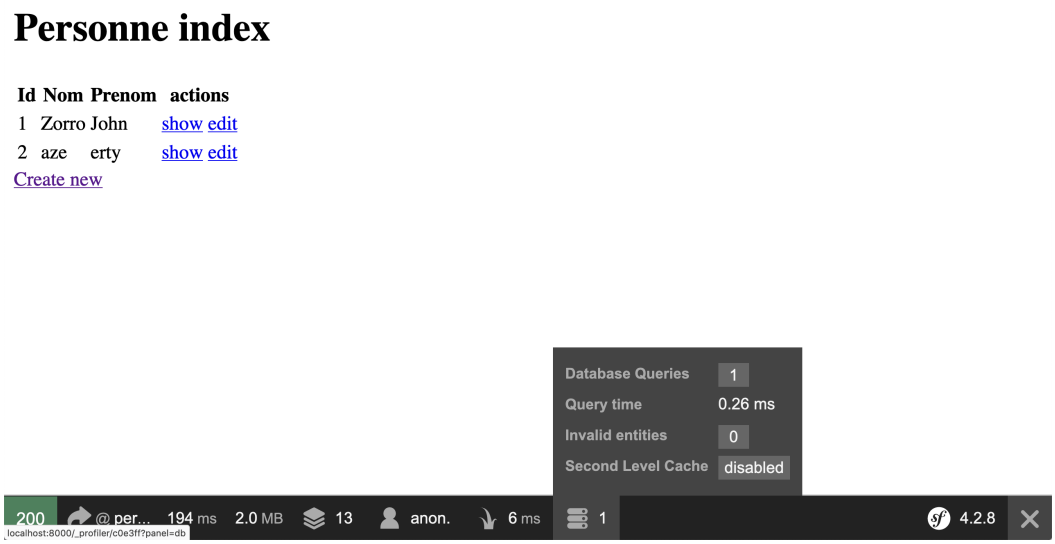
profiler

Puis installons la barre du profiler :

```
composer req profiler
```

Pour tester le CRUD et le profiler : localhost:8000/personne (http ://localhost :8000/personne)

On obtient :



debug des routes

On peut lister toutes les routes :

```
php bin/console debug:router
```

et obtenir en particulier :

Name	Method	Scheme	Host	Path
personne_index	GET	ANY	ANY	/personne/
personne_new	GET|POST	ANY	ANY	/personne/new
personne_show	GET	ANY	ANY	/personne/{id}
personne_edit	GET|POST	ANY	ANY	/personne/{id}/edit
personne_delete	DELETE	ANY	ANY	/personne//{id}
easyadmin	ANY	ANY	ANY	/admin

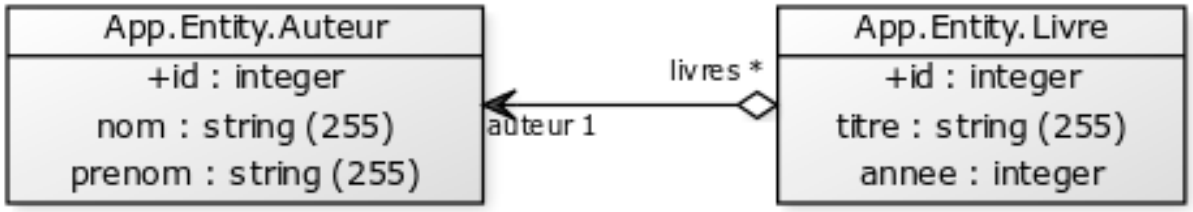
sécurité

On peut utiliser le security-checker pour vérifier tous les bundles installés :

```
composer require sec-checker --dev
```

3.37 Exemple Livres/Auteurs avec Doctrine

Vous pouvez reprendre ici une nouvelle application. On voudrait gérer une bibliothèque avec des Livres et des Auteurs. Les 2 entités se présentent ainsi :



3.37.1 Entités

Créons l'entité `Auteur`

```
php bin/console make:entity
```

— Répondez aux questions pour ajouter des champs nom et prenom de type string dans `Auteur`.
— Vérifiez la création du code correspondant dans src/Entity.

Puis faire de même avec l'entité `Livre`. Pour la relation, ajoutez un champ de type `relation` à l'entité `Livre`. Faites afficher toutes les possibilités avec le *?* :

```
Class name of the entity to create or update (e.g. AgreeableGnome):
> Livre

created: src/Entity/Livre.php
created: src/Repository/LivreRepository.php

Entity generated! Now let's add some fields!
You can always add more fields later manually or by re-running this␣
 ↪command.

New property name (press <return> to stop adding fields):
> titre

Field type (enter ? to see all types) [string]:
>

Field length [255]:
>

Can this field be null in the database (nullable) (yes/no) [no]:
>

updated: src/Entity/Livre.php

Add another property? Enter the property name (or press <return> to␣
 ↪stop adding fields):
> annee

Field type (enter ? to see all types) [string]:
```

(suite sur la page suivante)

(suite de la page précédente)

```
> ?

Main types
* string
* text
* boolean
* integer (or smallint, bigint)
* float

Relationships / Associations
* relation (a wizard  will help you build the relation)
* ManyToOne
* OneToMany
* ManyToMany
* OneToOne

Array/Object Types
* array (or simple_array)
* json
* object
* binary
* blob

Date/Time Types
* datetime (or datetime_immutable)
* datetimetz (or datetimetz_immutable)
* date (or date_immutable)
* time (or time_immutable)
* dateinterval

Other Types
* json_array
* decimal
* guid


Field type (enter ? to see all types) [string]:
> integer

Can this field be null in the database (nullable) (yes/no) [no]:
>

updated: src/Entity/Livre.php

Add another property? Enter the property name (or press <return> to␣
 ↪stop adding fields):
> auteur

Field type (enter ? to see all types) [string]:
```

(suite sur la page suivante)

(suite de la page précédente)

```
> relation

What class should this entity be related to?:
> Auteur

What type of relationship is this?
------------ ------------------------------------------------------
 ↪------------
Type         Description
------------ ------------------------------------------------------
 ↪------------
ManyToOne    Each Livre relates to (has) one Auteur.
             Each Auteur can relate to (can have) many Livre objects

OneToMany    Each Livre can relate to (can have) many Auteur␣
 ↪objects.
             Each Auteur relates to (has) one Livre

ManyToMany   Each Livre can relate to (can have) many Auteur␣
 ↪objects.
             Each Auteur can also relate to (can also have) many␣
 ↪Livre objects

OneToOne     Each Livre relates to (has) exactly one Auteur.
             Each Auteur also relates to (has) exactly one Livre.
------------ ------------------------------------------------------
 ↪------------

Relation type? [ManyToOne, OneToMany, ManyToMany, OneToOne]:
> ManyToOne

Is the Livre.auteur property allowed to be null (nullable)? (yes/
 ↪no) [yes]:
> no

Do you want to add a new property to Auteur so that you can access/
 ↪update Livre objects from it - e.g. $auteur->getLivres()? (yes/
 ↪no) [yes]:
> yes

A new property will also be added to the Auteur class so that you␣
 ↪can access the related Livre objects from it.

New field name inside Auteur [livres]:
>

Do you want to activate orphanRemoval on your relationship?
A Livre is "orphaned" when it is removed from its related Auteur.
e.g. $auteur->removeLivre($livre)
```

(suite sur la page suivante)

(suite de la page précédente)

```
NOTE: If a Livre may *change* from one Auteur to another, answer "no
 ↪".

Do you want to automatically delete orphaned App\Entity\Livre␣
 ↪objects (orphanRemoval)? (yes/no) [no]:
>   no
```

Voici l'entité `Livre` obtenue :

```php
<?php

   namespace App\Entity;

   use Doctrine\ORM\Mapping as ORM;

   /**
   * @ORM\Entity(repositoryClass="App\Repository\LivreRepository")
   */
   class Livre
   {
       /**
       * @ORM\Id()
       * @ORM\GeneratedValue()
       * @ORM\Column(type="integer")
       */
       private $id;

       /**
       * @ORM\Column(type="string", length=255)
       */
       private $titre;

       /**
       * @ORM\Column(type="integer")
       */
       private $annee;

       /**
       * @ORM\ManyToOne(targetEntity="App\Entity\Auteur",␣
↪inversedBy="livres")
       * @ORM\JoinColumn(nullable=false)
       */
       private $auteur;

       public function getId(): ?int
       {
           return $this->id;
       }
```

(suite sur la page suivante)

(suite de la page précédente)

```
    public function getTitre(): ?string
    {
        return $this->titre;
    }

    public function setTitre(string $titre): self
    {
        $this->titre = $titre;

        return $this;
    }

    public function getAnnee(): ?int
    {
        return $this->annee;
    }

    public function setAnnee(int $annee): self
    {
        $this->annee = $annee;

        return $this;
    }

    public function getAuteur(): ?Auteur
    {
        return $this->auteur;
    }

    public function setAuteur(?Auteur $auteur): self
    {
        $this->auteur = $auteur;

        return $this;
    }
}
```

3.37.2 Interaction avec la BD

On lance le makemigrations suivi du migrate :

```
php bin/console make:migration
php bin/console doctrine:migrations:migrate
```

En cas de problème, on peut forcer la synchonisation du nouveau schéma de BD :

```
./bin/console doctrine:schema:update --force
```

Voir la doc correspondante (https ://symfony.com/doc/current/doctrine/associations.html)

(Si *easy_admin* est installé, ajoutons un ou 2 auteurs puis livres dans easy_admin (http ://localhost :8000/admin))

3.37.3 Relançons le make :crud

```
php bin/console make:crud
```

pour les entités Auteur et Livre puis améliorons un peu les templates proposés.

3.37.4 Template de base avec Bootstrap

On peut déjà améliorer le template de base *base.html.twig* avec Bootstrap, ici *Bootswatch <https ://bootswatch.com/>* :

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>{% block title %}Bonjour{% endblock %}</title>
        {% block stylesheets %}
        <link rel="stylesheet" href="https://bootswatch.com/4/yeti/
↪bootstrap.min.css">
        {% endblock %}
    </head>
    <body>
<nav class="navbar navbar-expand-lg navbar-dark bg-primary">
<a class="navbar-brand" href="#">App de Gestion Livres/Auteurs</a>
<button class="navbar-toggler" type="button" data-toggle="collapse"␣
↪data-target="#navbarColor01" aria-controls="navbarColor01" aria-
↪expanded="false" aria-label="Toggle navigation">
    <span class="navbar-toggler-icon"></span>
</button>

<div class="collapse navbar-collapse" id="navbarColor01">
    <ul class="navbar-nav mr-auto">
    <li class="nav-item active">
        <a class="nav-link" href="#">Home <span class="sr-only">
↪(current)</span></a>
    </li>
    <li class="nav-item">
        <a class="nav-link" href="#">Livres</a>
    </li>
    <li class="nav-item">
```

(suite sur la page suivante)

(suite de la page précédente)

```
        <a class="nav-link" href="#">Auteurs</a>
    </li>
    <li class="nav-item">
        <a class="nav-link" href="#">A propos</a>
    </li>
    </ul>
</div>
</nav>
        <div class="container">
        {% block body %}
        {% endblock %}
        </div>
        {% block javascripts %}
    <script src="https://code.jquery.com/jquery-3.4.1.slim.min.js"
            integrity="sha256-
↪pasqAKBDmFT4eHoN2ndd6lN370kFiGUFyTiUHWhU7k8="
            crossorigin="anonymous">
        </script>
        <script src="https://bootswatch.com/_vendor/popper.js/dist/
↪umd/popper.min.js">
        </script>
        <script src="https://bootswatch.com/_vendor/bootstrap/dist/
↪js/bootstrap.min.js">
        </script>
        {% endblock %}
    </body>
</html>
```

on utilise les dépôts en ligne pour simplifier ici. Les liens sont à compléter . . .

Puis ensuite on fait hériter les autres templates de *base.html.twig* et on peut par exemple utiliser les boutons Bootstrap pour améliorer le rendu de la page *index.html.twig* des livres :

```
<button type="button" class="btn btn-info btn-sm">
<a href="{{ path('livre_show', {'id': livre.id}) }}">show</a>
</button>
<button type="button" class="btn btn-info btn-sm">
<a href="{{ path('livre_edit', {'id': livre.id}) }}">edit</a>
</button>
```

Si besoin, ajoutez une méthode *__toString()* à l'entité *Auteur* :

```
<?php
public function __toString()
{
    return $this->getPrenom() . '  ' . $this->getNom();
}
```

Faites de même dans l'entité *Livre*

Testez !

3.38 Utilisation de Faker pour saisir des données initiales dans la BD

Nous voudrions générer des données initiales automatiquement pour notre bibliothèque de Livres et Auteurs. Nous allons pour celà utiliser le Bundle Faker (https ://github.com/fzaninotto/Faker)

3.38.1 Installation de DoctrineFixturesBundle

```
composer req --dev make doctrine/doctrine-fixtures-bundle
```

Ceci crée dans le dossier *src* un dossier *DataFixtures* contenant un fichier à compléter *AppFixtures.php*. On peut utiliser *DoctrineFixturesBundle* sans Faker mais pas aussi convivial !

Voir la doc correspondante (https ://symfony.com/doc/current/bundles/DoctrineFixturesBundle/index.html)

3.38.2 Installation de Faker

Le Bundle Faker est spécialisé dans la génération de données aléatoires vraisemblables de tous types (chaines, noms, adresses, lorem, nombres, dates, etc.) avec localisation.

Nous l'installons avec :

```
composer require --dev fzaninotto/faker
```

3.38.3 Complétons AppFixtures.php

Nous allons maintenant compléter le fichier AppFixtures.php pour qu'il crée automatiquement un ensemble de données initiales de Livres et d'Auteurs.

```
<?php

  namespace App\DataFixtures;

  use App\Entity\Livre;
  use App\Entity\Auteur;

  use Doctrine\Bundle\FixturesBundle\Fixture;
  use Doctrine\Common\Persistence\ObjectManager;
  use Faker;

 class AppFixtures extends Fixture
 {
     public function load(ObjectManager $manager)
      {
```

(suite sur la page suivante)

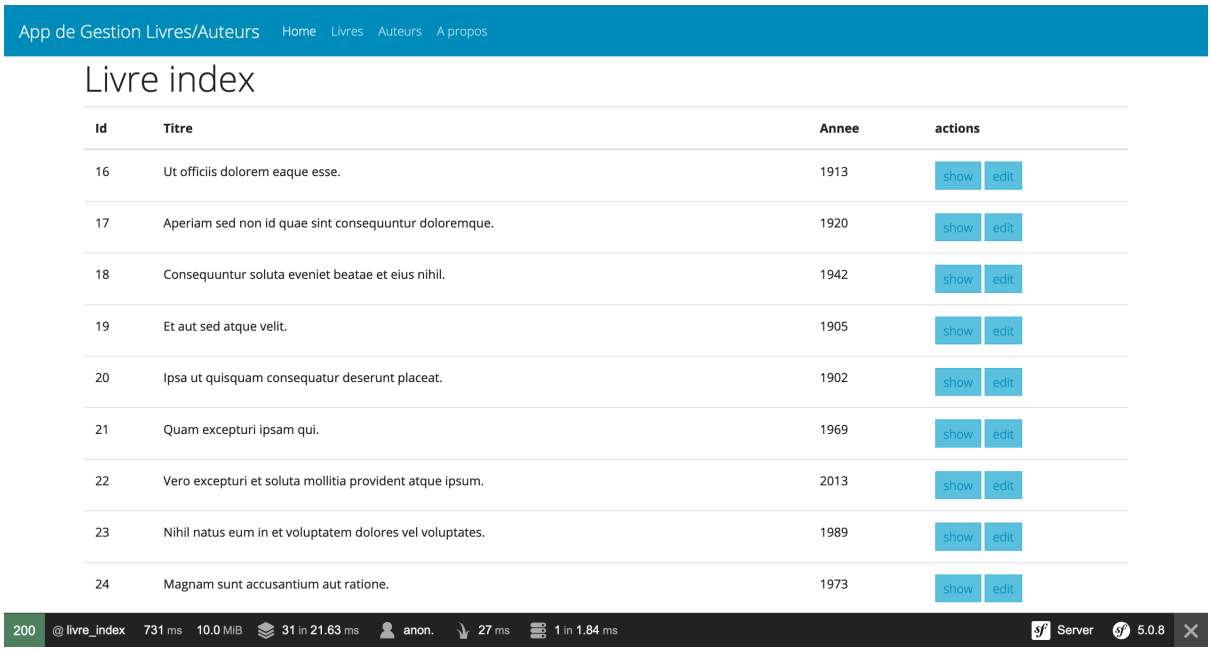
(suite de la page précédente)

```
        $faker = Faker\Factory::create('fr_FR');
            // on crée 4 auteurs avec noms et prénoms "aléatoires"␣
↪en français
            $auteurs = Array();
            for ($i = 0; $i < 4; $i++) {
                $auteurs[$i] = new Auteur();
                $auteurs[$i]->setNom($faker->lastName);
                $auteurs[$i]->setPrenom($faker->firstName);

                $manager->persist($auteurs[$i]);
            }
        // nouvelle boucle pour créer des livres

        $livres = Array();

        for ($i = 0; $i < 12; $i++) {
                $livres[$i] = new Livre();
                $livres[$i]->setTitre($faker->sentence($nbWords = 6,
↪$variableNbWords = true));
                $livres[$i]->setAnnee($faker->numberBetween($min =␣
↪1900, $max = 2020));
                $livres[$i]->setAuteur($auteurs[$i % 3]);

                $manager->persist($livres[$i]);
            }

            $manager->flush();
        }
    }
```

3.38.4 Chargeons les fixtures

```
php bin/console doctrine:fixtures:load
```

Testons :



3.39 API Livres/Auteurs

Pour fabriquer une API nous pouvons le faire :

— à la Main
— avec APIPlatform

3.39.1 API livres et auteurs « à la main »

Avant de commencer nous allons installer le bundle nelmio/cors-bundle :

```
composer req cors
```

ou

```
composer require nelmio/cors-bundle
```

L'installation se lance.

```
Using version ^2.0 for nelmio/cors-bundle
./composer.json has been updated
Loading composer repositories with package information
Updating dependencies (including require-dev)
Restricting packages listed in "symfony/symfony" to "5.0.*"
Package operations: 1 install, 0 updates, 0 removals
- Installing nelmio/cors-bundle (2.0.1): Downloading (100%)
Writing lock file
Generating autoload files
```

(suite sur la page suivante)

(suite de la page précédente)

```
ocramius/package-versions: Generating version class...
ocramius/package-versions: ...done generating version class
Symfony operations: 1 recipe (b3d83b533e25e4f0b71e3e2b85c51c5d)
- Configuring nelmio/cors-bundle (>=1.5): From github.com/symfony/
 ↪recipes:master
Executing script cache:clear [OK]
Executing script assets:install public [OK]

Some files may have been created or updated to configure your new␣
 ↪packages.
Please review, edit and commit them: these files are yours.
```

Le cors-bundle permet de définir les règles CORS. Ce bundle permet également de définir les domaines qui auront accès à votre API REST.

Vérifions que les règles ont bien été définies dans /config/packages/nelmio_cors.yaml

```
nelmio_cors:
    defaults:
        origin_regex: true
        allow_origin: ['%env(CORS_ALLOW_ORIGIN)%']
        allow_methods: ['GET', 'OPTIONS', 'POST', 'PUT', 'PATCH',
 ↪'DELETE']
        allow_headers: ['Content-Type', 'Authorization']
        expose_headers: ['Link']
        max_age: 3600
    paths:
        '^/': null
```

Dans .env modifions le paramètre d'autorisation des domaines.

```
###> nelmio/cors-bundle ###
CORS_ALLOW_ORIGIN=*
###< nelmio/cors-bundle ###
```

Ce paramètre permet à n'importe quel domaine d'accéder à notre API. /!\ A n'utiliser que pour des APIs puliques /!

Ajoutons un controleur d'Auteur et un autre de Livre avec :

```
php bin/console make:controller AuteurController --no-template
php bin/console make:controller LivreController --no-template
```

On obtient déjà par exemple pour `AuteurController` un index qui renvoie un contenu json :

```
<?php

   namespace App\Controller;
```

(suite sur la page suivante)

(suite de la page précédente)

```
use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\Routing\Annotation\Route;

class AuteurController extends AbstractController
{
    /**
    * @Route("/auteur", name="auteur")
    */
    public function index()
    {
        return $this->json([
            'message' => 'Welcome to your new controller!',
            'path' => 'src/Controller/AuteurController.php',
        ]);
    }
}
```

Lister tous les auteurs

Complétons `AuteurController` :

```
<?php

namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\Routing\Annotation\Route;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\HttpFoundation\Request;
use App\Entity\Auteur;

/**
* @Route("/books/api/v1.0")
*/
class AuteurController extends AbstractController
{

    /**
    * Permet d'avoir la liste de tous les auteurs
    * @Route("/auteur", name="liste_auteur", methods={"GET"})
    */
    public function listeAuteur()
    {
        $repository   = $this->getDoctrine()->
↪getRepository(Auteur::class);
        $listeAuteur  = $repository->findAll();
        $listeReponse = array();
        foreach ($listeAuteur as $auteur) {
```

(suite sur la page suivante)

(suite de la page précédente)

```
                $listeReponse[] = array(
                'id'     => $auteur->getId(),
                'nom'    => $auteur->getNom(),
                'prenom' => $auteur->getPrenom(),
            );
        }
        $reponse = new Response();
        $reponse->setContent(json_encode(array("auteur"=>
↪$listeReponse)));
        $reponse->headers->set("Content-Type", "application/json");
        $reponse->headers->set("Access-Control-Allow-Origin", "*");
        return $reponse;
    }
```

Puis pour avoir les détails d'un auteur

```
<?php
/**
 * Permet d'avoir les livre d'un auteur grâce à son id
 * @Route("/auteur/{id}", name="details_auteur", methods={"GET"})
 */
public function detailsAuteur($id)
{
    $repository = $this->getDoctrine()->
↪getRepository(Auteur::class);
    $auteur     = $repository->find($id);
    $listeLivre = $auteur->getLivres();
    $livres = [];
    foreach ($listeLivre as $livre) {
        $livres[] = array(
            "id"    => $livre->getId(),
            "titre" => $livre->getTitre(),
        );
    }
    $reponse = new Response(json_encode(array(
                    'id'     => $auteur->getId(),
                    'nom'    => $auteur->getNom(),
                    'prenom' => $auteur->getPrenom(),
                    'livres' => $livres,
                    ))
            );
    $reponse->headers->set("Content-Type", "application/json");
    $reponse->headers->set("Access-Control-Allow-Origin", "*");
    return $reponse;
}
```

Créer un auteur

```php
<?php
/**
 * Permet de créer un auteur
 * @Route("/auteur", name="nouveau_auteur", methods={"POST"})
 */
public function nouveauAuteur(Request $request)
{
    $entityManager = $this->getDoctrine()->getManager();
    $auteur = new Auteur();
    $body   = json_decode($request->getContent(), true);
    $nom    = $body['nom'];
    $prenom = $body['prenom'];
    $auteur->setNom($nom);
    $auteur->setPrenom($prenom);
    $entityManager->persist($auteur);
            $entityManager->flush();

    $reponse = new Response(json_encode(array(
                    'id'     => $auteur->getId(),
                    'nom'    => $auteur->getNom(),
                    'prenom' => $auteur->getPrenom()
                    )
            ));

    $reponse->headers->set("Content-Type", "application/json");
    $reponse->headers->set("Access-Control-Allow-Origin", "*");
    return $reponse;
}
```

Supprimer un auteur

```php
<?php
/**
 * Permet de supprimer un auteur grâce à son id
 * @Route("/auteur", name="suppression_auteur", methods={"DELETE"})
 */
public function suppressionAuteur(Request $request)
{
    $entityManager = $this->getDoctrine()->getManager();
    $repository    = $this->getDoctrine()->
↪getRepository(Auteur::class);
    $body          = json_decode($request->getContent(), true);
    $id            = $body['id'];
    $auteur        = $repository->find($id);
    $entityManager->remove($auteur);
            $entityManager->flush();
```

(suite sur la page suivante)

(suite de la page précédente)

```
            $reponse = new Response(json_encode(array(
                    'nom'    => $auteur->getNom(),
                    'prenom' => $auteur->getPrenom(),
                    ))
            );
    $reponse->headers->set("Content-Type", "application/json");
    $reponse->headers->set("Access-Control-Allow-Origin", "*");
    return $reponse;
}
```

Modifier un auteur

```
<?php
/**
 * Permet de modifier le nom et/ou le prenom d'un auteur grâce à␣
↪son id
 * La gestion des livres de l'auteur se fera via l'entité livre
 * @Route("/auteur", name="modification_auteur", methods={"PUT"})
 */
public function modificationAuteur(Request $request)
{
    $entityManager = $this->getDoctrine()->getManager();
    $repository    = $this->getDoctrine()->
↪getRepository(Auteur::class);
    $body          = json_decode($request->getContent(), true);
    $id            = $body['id'];
    $nom           = $body['nom'];
    $prenom        = $body['prenom'];
    $auteur        = $repository->find($id);
    $auteur->setNom($nom);
    $auteur->setPrenom($prenom);
    $entityManager->persist($auteur);
            $entityManager->flush();
            $reponse = new Response(json_encode(array(
                        'id'     => $auteur->getId(),
                    'nom'    => $auteur->getNom(),
                    'prenom' => $auteur->getPrenom(),
                    ))
            );
    $reponse->headers->set("Content-Type", "application/json");
    $reponse->headers->set("Access-Control-Allow-Origin", "*");
    return $reponse;
}
```

L'API/CRUD de l'entité Auteur est terminée. Faites les mêmes manipulations pour l'entité Livre : lister tous les livres, détails d'un livre, créer un livre, supprimer un livre et modifier un livre. N'oubliez pas le livre stock l'auteur.

Notre API est complète mais nous avons du travailler dur ... Nous aurions pu prendre une solution de facilité !

3.39.2 Avec le composant d'API APIPlatform

Installation d'APIPlatform

On l'installe avec la recette flex correspondante :

```
composer req api
```

Annotons les entités `Auteur` et `Livres` en ajoutant l'import :

```
<?php

    use ApiPlatform\Core\Annotation\ApiResource;
```

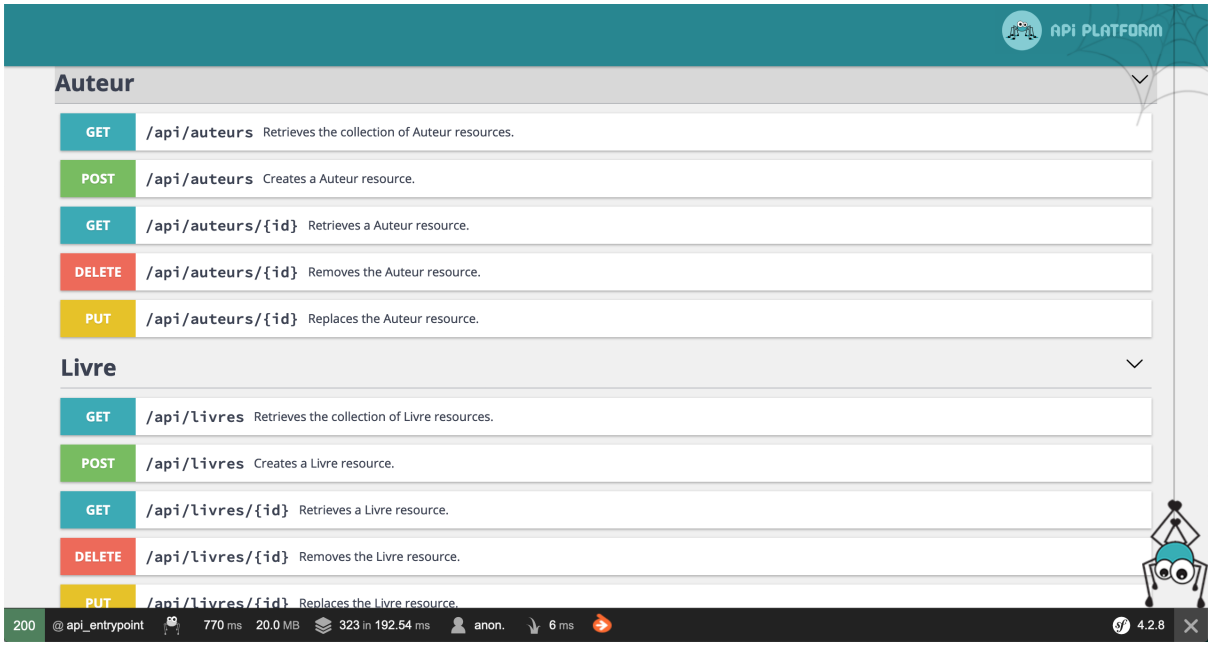
et l'annotation :

```
/**
* @ApiResource()
*/
```

Voilà !

Visitons la page : API (http ://localhost :8000/api)

Notre API est prête à l'emploi :



GraphQL

GraphQL permet de faire des requêtes plus complexes sur une API et permet donc d'interopérer de façon plus complète avec cette API. Voir GraphQL (https ://graphql.org/learn/) .

Pour activer le support GraphQL dans Symfony :

```
composer require webonyx/graphql-php,
```

Puis visiter : /api/graphql (http ://localhost :8000/api/graphql)

Consultez la documentation d'"APIPlatform (https ://api-platform.com/docs)

FriendsOfSymfony REST Bundle

Une autre solution pour fabriquer un service REST est de passer par un autre Bundle connu : FOS (Friends Of Symfony). On installe le Bundle REST correspondant avec :

```
composer require friendsofsymfony/rest-bundle
```

Un exemple complet se trouve sur : Symfony5ApiRest (https ://github.com/Tony133/Symfony5ApiRest)

3.40 Consultation de l'API avec JS : fetch, await, async

Il nous reste à consulter cette API en JS. Nous pouvons utiliser jQuery, des composants Web ou plus directement :

- fetch
- await
- async :

3.40.1 Interrogation de l'API auteurs

Voici quelques méthodes JS pour dialoguer avec cette API.

```
//liste tous les auteurs de la base
async function getAuteurs(){
    let rep = await fetch('http://localhost:8000//books/api/v1.0/
↪auteur', { method: 'GET' });
    let reponse = await rep.json();
    return reponse;
}

//donne les détails d'un auteur par son id
async function auteurById(id){
    let rep = await fetch('http://localhost:8000/books/api/v1.0/
↪auteur/'+id, { method: 'GET' });
```

(suite sur la page suivante)

(suite de la page précédente)

```
    let reponse = await rep.json();
    return reponse;
}


//supprime un auteur grâce à son id
async function auteurSupprime(id){
    let rep = await fetch('http://localhost:8000/books/api/v1.0/
↪auteur/',
     { method: 'DELETE', body: JSON.stringify({ "id": id })
    });
    let reponse = await rep.json();
    return reponse;
}
```

3.40.2 Interrogation de l'API livres

```
// liste tous les livres
async function getLivres(){
    let rep = await fetch('http://localhost:8000/books/api/v1.0/
↪livres/', { method: 'GET' });
    let reponse = await rep.json();
    return reponse;
}

//donne les détails d'un livre par id
async function livreById(id){
    let rep = await fetch('http://localhost:8000/books/api/v1.0/
↪livres/'+id, { method: 'GET' });
    let reponse = await rep.json();
    return reponse;
}


//supprime un livre grâce à son id
async function livreSupprime(id){
    let rep = await fetch('http://localhost:8000/books/api/v1.0/
↪livre',
     { method: 'DELETE', body: JSON.stringify({ "id": id })
    });
    let reponse = await rep.json();
    return reponse;
}
```

Il vous reste à compléter les méthodes `POST` et `PUT` pour ces 2 entités et à intégrer ce code dans un joli composant *Front*.

3.41 Authentification élémentaire en Symfony

Nous allons ici présenter comment faire une authentification « à la main » en Symfony. Il existe d'autres méthodes plus éprouvées de réaliser cette fonctionnalité mais il est bon dans sa progression en PHP/Symfony de passer par cette étape.

3.41.1 Initialisation du projet Symfony

Tout d'abord création d'un nouveau projet Symfony (code pour la v5 de Symfony)

```
symfony new --full my_project
cd my_project
```

Ensuite nous allons créer une entité utilisateur grâce au makerbundle de Symfony

```
php bin/console make:entity
```

Les attributs de base pour notre utilisateur sont pseudo, motDePasse et role. Ce dernier nous permettra de définir si l'utilisateur dispose de droits spéciaux (ex : admin). Vous pouvez bien evidemment ajouter tous les attributs nécessaires à votre application.

```
netrestore-3c0754698637:authentification utilisateur$ php bin/console make:entity

  Class name of the entity to create or update (e.g. AgreeableGnome):
 > Utilisateur

  Your entity already exists! So let's add some new fields!

  New property name (press <return> to stop adding fields):
  > pseudo

  Field type (enter ? to see all types) [string]:
 >

  Field length [255]:
  >

  Can this field be null in the database (nullable) (yes/no) [no]:
  >

  updated: src/Entity/Utilisateur.php

  Add another property? Enter the property name (or press <return> to stop adding fields):
  > motDePasse

  Field type (enter ? to see all types) [string]:
 >

  Field length [255]:
  >

  Can this field be null in the database (nullable) (yes/no) [no]:
  >

  updated: src/Entity/Utilisateur.php

  Add another property? Enter the property name (or press <return> to stop adding fields):
  > role

  Field type (enter ? to see all types) [string]:
 >

  Field length [255]:
  >

  Can this field be null in the database (nullable) (yes/no) [no]:
  >

  updated: src/Entity/Utilisateur.php

  Add another property? Enter the property name (or press <return> to stop adding fields):
  >


  Success!

  Next: When you're ready, create a migration with php bin/console make:migration
```

Enfin nous créons un CRUD pour notre entité Utilisateur

```
php bin/console make:crud
```

Cette commande nous permet d'avoir tous les outils nécessaires pour la bonne gestion de nos utilisa

— un controller implémentant les fonctionnalités nécessaires
— l'utilisation des formulaires Symfony

— des templates qui ne demandent qu'à être modifiés

Bien sûr nous pourrions tout faire à la main, mais Symfony nous permet de gagner du temps. Faites la migration.

3.41.2 Modification de notre application

L'application telle que nous l'avons, ne nous permet pas de gérer l'authentification de manière adéquate. Pour se faire, il va nous falloir apporter quelques modifications.

La création d'un utilisateur

Pour commencer, la méthode new :

```
<?php
    /**
     * @Route("/new", name="utilisateur_new", methods={"GET","POST"}
↪)
     */
    public function new(Request $request, UtilisateurRepository
↪$utilisateurRepository): Response
    {
        $utilisateur = new Utilisateur();
        $form = $this->createForm(UtilisateurType::class,
↪$utilisateur);
        $form->handleRequest($request);

        if ($form->isSubmitted() && $form->isValid()) {
            //est ce que le pseudo est unique
            //utilisation d'une méthode que nous allons ajouter au␣
↪repository
            if($utilisateurRepository->findOneByPseudo($utilisateur-
↪>getPseudo()) != null)
            {// s'il ne l'est pas, on renvoie vers la page de␣
↪création d'utilisateur avec une notification
                return $this->render('utilisateur/new.html.twig', [
                    'utilisateur' => $utilisateur,
                    'form'        => $form->createView(),
                    'message'     => "Ce pseudo est déjà utilisé,␣
↪merci d'en changer"
                ]);
            }
            $entityManager = $this->getDoctrine()->getManager();
            //Nous modifions l'utilisateur
            $pass = password_hash($utilisateur->getMotDePasse(),␣
↪PASSWORD_DEFAULT);
            $utilisateur->setMotDePasse($pass);//mot de passe crypté
            $utilisateur->setRole("Utilisateur");//par défaut un␣
↪nouvel utilisateur aura un role classique
```

(suite sur la page suivante)

(suite de la page précédente)

```
            $entityManager->persist($utilisateur);
            $entityManager->flush();

            return $this->redirectToRoute('index');//on ne veut pas␣
↪que l'utilisateur ai accès à la liste de tous les autres␣
↪utilisateurs
        }

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

A la base, cette méthode ajoute un utilisateur, mais nous souhaitons tout d'abord que le pseudo soit unique, ensuite que le mot de passe soit chiffré

Danger : /!\ NE JAMAIS STOCKER DE MOT DE PASSE EN CLAIR /!\

et enfin attribuer un role par défaut.

Création de la méthode *findOneByPseudo* dans UtilisateurRepository

```
<?php
    /**
     * Cette méthode va nous permettre de récupérer un utilisateur␣
↪via son pseudo
     */
    public function findOneByPseudo($value): ?Utilisateur
    {
        return $this->createQueryBuilder('u')
            ->andWhere('u.pseudo = :val')
            ->setParameter('val', $value)
            ->getQuery()
            ->getOneOrNullResult()
        ;
    }
```

Avant de tester, nous allons créer une route index, puis une dans notre template de création dans le cas où le pseudo est déjà utilisé.

Route index :

```
<?php
    /**
     * @Route("/", name="index", methods={"GET"})
     * Nous la modifierons plus tard
     */
```

(suite sur la page suivante)

(suite de la page précédente)

```
    public function page_index(UtilisateurRepository
↪$utilisateurRepository): Response
    {
        return $this->render('utilisateur/index.html.twig', [
            'utilisateurs' => $utilisateurRepository->findAll(),
        ]);
    }
```

Base template :

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>{% block title %}Welcome!{% endblock %}</title>
        {% block stylesheets %}
        <link rel="stylesheet" href="https://bootswatch.com/4/yeti/
↪bootstrap.min.css">
        {% endblock %}
    </head>
    <body>
        {% block body %}{% endblock %}
        {% block javascripts %}{% endblock %}
    </body>
</html>
```

On pourra utiliser les classes Bootstrap pour faire un CSS agréable et facile à mettre en place.

Template new :

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

{% block title %}New Utilisateur{% endblock %}

{% block body %}
    <h1>Create new Utilisateur</h1>

    {# nous testons si la variable message existe #}
    {% if message is defined %}
        <div class="alert alert-danger">
            {{ message }}
        </div>
    {% endif %}

    {{ include('utilisateur/_form.html.twig') }}

    <a href="{{ path('utilisateur_index') }}">back to list</a>
{% endblock %}
```

Nous pouvons maintenant tester notre modification :

— démarrer le serveur
— https://127.0.0.1:8000/utilisateur/new
— saisir un premier utilisateur
— https://127.0.0.1:8000/utilisateur/new
— saisir un second utilisateur avec le même pseudo que le premier.

Une fois que tout est vérifié, passons à la suite.

Le mot de passe apparait en clair dans le formulaire et nous devons mettre en place un test sur le mot de passe en le faisant saisir une seconde fois.

Cacher le mot de passe (/Form/UtilisateurType) :

```
<?php

    namespace App\Form;

    use App\Entity\Utilisateur;
    use Symfony\Component\Form\AbstractType;
    use Symfony\Component\Form\FormBuilderInterface;
    use Symfony\Component\OptionsResolver\OptionsResolver;
    //ajout du use pour utiliser le type input password de Symfony
    use Symfony\Component\Form\Extension\Core\Type\PasswordType;

    class UtilisateurType extends AbstractType
    {
        public function buildForm(FormBuilderInterface $builder,␣
↪array $options)
        {
            $builder
                ->add('pseudo')
                ->add('motDePasse', PasswordType::class)//input␣
↪type=text devient input type=password
                //suppression de la possibilité d'ajouter un rôle
            ;
        }

        public function configureOptions(OptionsResolver $resolver)
        {
            $resolver->setDefaults([
                'data_class' => Utilisateur::class,
            ]);
        }
    }
```

Nous modifions directement dans le formbuilder, cela permettra à notre modification de se propager dans toute l'application.

Nous modifions maintenant le template du new afin d'ajouter l'input de vérification du mot de passe :

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

{% block title %}New Utilisateur{% endblock %}

{% block body %}
    <h1>Create new Utilisateur</h1>

    {# nous testons si la variable message existe #}
    {% if message is defined %}
        <div class="alert alert-danger">
            {{ message }}
        </div>
    {% endif %}

    {{ form_start(form, {'attr': {'id': 'new_edit_utilisateur'}}) }}
    {# utilisation de classes bootstrap pour la mise en forme #}
        <div class="row">
            <div class="col-12">
                {{ form_label(form.pseudo) }}
                {{ form_widget(form.pseudo) }}
            </div>
            <div class="col-12">
                {{ form_label(form.motDePasse) }}
                {{ form_widget(form.motDePasse) }}
            </div>
            <div class="col-12">
                <label for="verifpass">Saisir une seconde fois le␣
↪mot de passe</label>
                <input type="password" id="verifpass" required>
            </div>
        </div>
        <button class="btn btn-success">{{ button_label|default(
↪'Save') }}</button>
    {{ form_end(form) }}

    <a href="{{ path('utilisateur_index') }}">back to list</a>
{% endblock %}
```

Création d'un script JS avec jQuery afin de tester si les deux mot de passe sont identiques. Créez tout d'abord un dossier js dans public puis dans ce dossier, un fichier script.js.

Script de vérification :

```
$("#new_edit_utilisateur").on('submit', function(){
    if($("#utilisateur_motDePasse").val() != $("#verifpass").val())
↪{
        //implémntez votre code
        alert("Les deux mots de passe saisies sont différents");
        alert("Merci de renouveler l'opération");
        return false;
```

(suite sur la page suivante)

(suite de la page précédente)

```
    }
})
```

Il ne faut pas oublier d'intégrer le lien vers jQuery et notre script.js dans le base template :

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>{% block title %}Welcome!{% endblock %}</title>
        {% block stylesheets %}
        <link rel="stylesheet" href="https://bootswatch.com/4/yeti/
↪bootstrap.min.css">
        {% endblock %}
    </head>
    <body>
        {% block body %}{% endblock %}
        {% block javascripts %}
            <script
                src="https://code.jquery.com/jquery-3.5.1.min.js"
                integrity="sha256-9/aliU8dGd2tb6OSsuzixeV4y/
↪faTqgFtohetphbbj0="
                crossorigin="anonymous">
            </script>
            <script src="/js/script.js"></script>
        {% endblock %}
    </body>
</html>
```

Nous avons fini la modification pour la création d'un utilisateur.

3.41.3 Création de la page de connexion

Maintenant que nous avons le code pour créer l'utilisateur, nous allons passer à la partie concernant la connexion.

Template de connexion

Créer pour cela un nouveau template connexion.html.twig dans template/utilisateur :

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

{% block title %}Connexion{% endblock %}

{% block body %}
    <h1>Se connecter</h1>
```

(suite sur la page suivante)

(suite de la page précédente)

```
    {# nous testons si la variable message existe #}
    {% if message is defined %}
        <div class="alert alert-danger">
            {{ message }}
        </div>
    {% endif %}

    {{ include('utilisateur/_form.html.twig', {'button_label': 'Se␣
↪connecter'}) }}

    <a href="{{ path('utilisateur_index') }}">back to list</a>
{% endblock %}
```

Côté controller

```
<?php
    /**
    * @Route("/connexion", name="connexion", methods={"GET", "POST"}
↪)
    */
    public function connexion(Request $request,␣
↪UtilisateurRepository $utilisateurRepository, Session $session):␣
↪Response
    {
        //en cas de connexion ouverte
        if($session->has('user'))
        {
            //on la referme, ain de pouvoir initier une nouvelle␣
↪connexion
            $session->remove('user');
        }

        $utilisateur = new Utilisateur();
        $form = $this->createForm(UtilisateurType::class,
↪$utilisateur);
        $form->handleRequest($request);

        if ($form->isSubmitted() && $form->isValid()) {
            $pseudo      = $utilisateur->getPseudo();
            //on récupère le code crypté
            $passHash    = $utilisateurRepository->findOneByPseudo(
↪$pseudo)->getMotDePasse() ?? "pas d'utilisateur";
            //cette méthode vérifie que le mot de passe saisie et␣
↪le hash correspondent
            $password    = password_verify($utilisateur->
↪getMotDePasse(), $passHash);
            if($password)
```

(suite sur la page suivante)

(suite de la page précédente)

```
            {
                $utilisateur = $utilisateurRepository->
↪findOneByPseudo($pseudo);
                //on ouvre la connexion
                $session->set('user', $utilisateur);
                return $this->redirectToRoute('index');
            }

            return $this->render('utilisateur/connexion.html.twig',␣
↪[
                'form'    => $form->createView(),
                'message' => "Connexion refusée"
                ]);
        }
            return $this->render('utilisateur/connexion.html.twig',␣
↪[
            'form'    => $form->createView()
            ]);
    }
```

Nous utilisons l'objet Session de HttpFoundation, cela va nous permettre de maintenir ouverte une session et donc de savoir si un utilisateur est connecté. Ne pas oublier use Symfony\Component\HttpFoundation\Session\Session ; dans le controller.

Pour des raisons de facilité, nous fermons la connexion dès que l'utilisateur souhaite accéder à la page de connexion. En temps normal nous préférerons un bouton qui permet à l'utilisateur de fermer sa connexion. Le but de ce tuto n'est pas de tout faire, mais juste d'exposer les principes et la manière d'y arriver.

A vous de vous appuyer sur ce tuto pour développer votre propre interface.

3.41.4 Naviguer en utilisant l'authentification

Nous allons maintenant voir comment naviguer dans notre application. Il est possible que l'on souhaite garder secrêtes des pages pour les inscrits, ou accessible que pour les administrateurs, etc . . .

Navigation avec les droits

Dans mon exemple je veux que seul un admin puisse avoir le droit d'aller voir la liste des utilisateurs.

```
<?php
    /**
    * @Route("/", name="utilisateur_index", methods={"GET"})
    */
    public function index(UtilisateurRepository
↪$utilisateurRepository, Session $session): Response
```

(suite sur la page suivante)

(suite de la page précédente)

```
    {
        if(!$session->has('user') || $session->get('user')->
↪getRole() != 'Admin')
        {
            //Permet de notifier l'utilisateur
            $session->set("message", "Vous n'avez pas le droit d
↪'acceder à la page admin vous avez été redirigé sur cette page");
            return $this->redirectToRoute('index');
        }
        return $this->render('utilisateur/index.html.twig', [
            'utilisateurs' => $utilisateurRepository->findAll(),
        ]);
    }
```

Dans le cas où il s'agit d'un simple utilisateur ou d'une personne non connectée, on redirige vers une autre page en notifiant l'utilisateur.

Dans le controller :

```
<?php
    /**
    * @Route("/page", name="index", methods={"GET"})
    * Page dédiée à tous les utilisateurs inscrit ou non
    */
    public function page_index(UtilisateurRepository
↪$utilisateurRepository, Session $session): Response
    {
        $btn     = "Connexion"; //label du bouton
        $class   = "success"; //couleur du bouton

        if($session->has('user'))
        {
            $btn   = "Déconnexion";
            $class = "danger";
        }

        $return  = array( //initialisation de l'array de paramètre␣
↪afin d'ajouter ou non un message
            'btn'           => $btn,
            'class'         => $class
        );

        if($session->has('message'))
        {
            $message = $session->get('message');
            $session->remove('message'); //on vide la variable␣
↪message dans la session
            $return['message'] = $message; //on ajoute à l'array de␣
↪paramètres notre message
```

(suite sur la page suivante)

(suite de la page précédente)

```
        }

        return $this->render('utilisateur/page-utilisateur.html.twig
↪', $return);
    }
```

Côté template (dans un nouveau template /utilisateur/page-utilisateur.html.twig) :

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

{% block title %}Page utilisateur{% endblock %}

{% block body %}
    <h1>Page ouverte aux utilisateurs et admin</h1>

    <a href="{{ path('utilisateur_index') }}" class="btn btn-warning
↪">Page admin</a>
    {# nous testons si la variable message existe #}
    {% if message is defined %}
        <div class="alert alert-danger">
            {{ message }}
        </div>
    {% endif %}

    <p>
    Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed␣
↪non risus. Suspendisse lectus tortor, dignissim sit amet,
    adipiscing nec, ultricies sed, dolor. Cras elementum ultrices␣
↪diam. Maecenas ligula massa, varius a, semper congue, euismod non,
↪ mi.
    Proin porttitor, orci nec nonummy molestie, enim est eleifend␣
↪mi, non fermentum diam nisl sit amet erat. Duis semper. Duis arcu␣
↪massa,
    scelerisque vitae, consequat in, pretium a, enim. Pellentesque␣
↪congue. Ut in risus volutpat libero pharetra tempor. Cras␣
↪vestibulum bibendum augue.
    Praesent egestas leo in pede. Praesent blandit odio eu enim.␣
↪Pellentesque sed dui ut augue blandit sodales. Vestibulum ante␣
↪ipsum primis in
    faucibus orci luctus et ultrices posuere cubilia Curae; Aliquam␣
↪nibh. Mauris ac mauris sed pede pellentesque fermentum. Maecenas␣
↪adipiscing ante non diam sodales hendrerit.

    Aliquam convallis sollicitudin purus. Praesent aliquam, enim at␣
↪fermentum mollis, ligula massa adipiscing nisl, ac euismod nibh␣
↪nisl eu lectus.
    Fusce vulputate sem at sapien. Vivamus leo. Aliquam euismod␣
↪libero eu enim. Nulla nec felis sed leo placerat imperdiet.␣
↪Aenean suscipit nulla in justo. Suspendisse cursus rutrum augue.
```

(suite sur la page suivante)

(suite de la page précédente)

```
    Nulla tincidunt tincidunt mi. Curabitur iaculis, lorem vel␣
↪rhoncus faucibus, felis magna fermentum augue, et ultricies lacus␣
↪lorem varius purus. Curabitur eu amet.
    </p>

    <a href="{{ path('connexion') }}" class="btn btn-{{class}}">{{␣
↪btn }}</a>
    <a href="{{ path('utilisateur_new') }}" class="btn btn-primary">
↪S'enregistrer</a>
{% endblock %}
```

Il ne reste plus qu'a créer un utilisateur admin pour tester. Afin de tester un utilisateur admin, vous pouvez changer le rôle par défaut dans la méthode de création afin de créer un utilisateur avec le rôle Admin. Une fois que vous vous serez reconnecté avec votre compte admin, vous aurez accès à la page admin.

Avertissement : Méthodes d'authentification

Pour rappel ce tuto est fait pour exposer les principes de l'authentification à la main, il ne fait en aucun cas la présentation d'une authentification complète mais met à disposition quelques outils pour y parvenir . . .

A vous de jouer !

3.42 Authentification avec le SecurityBundle de Symfony

Nous allons ici présenter comment faire une authentification en utilisant le *SecurityBundle de Symfony* . Il permet la connexion et la navigation de manière sécurisée.

3.42.1 Initialisation du projet Symfony

Tout d'abord création d'un nouveau projet Symfony (code pour la v5 de Symfony)

```
symfony new --full my_project
cd my_project
```

3.42.2 Installation de SecurityBundle

```
composer require symfony/security-bundle
```

3.42.3 Création de l'entité utilisateur

```
php bin/console make:user
```

```
netrestore-3c0754698637:securitybundle utilisateur$ php bin/console make:user

 The name of the security user class (e.g. User) [User]:
 > Utilisateur

 Do you want to store user data in the database (via Doctrine)? (yes/no) [yes]:
 >

 Enter a property name that will be the unique "display" name for the user (e.g. email, username, uuid) [email]:
 > username

 Will this app need to hash/check user passwords? Choose No if passwords are not needed or will be checked/hashed

 Does this app need to hash/check user passwords? (yes/no) [yes]:
 > yes

 created: src/Entity/Utilisateur.php
 created: src/Repository/UtilisateurRepository.php
 updated: src/Entity/Utilisateur.php
 updated: config/packages/security.yaml

 [OK] Success!

 Next Steps:
   - Review your new App\Entity\Utilisateur class.
   - Use make:entity to add more fields to your Utilisateur entity and then run make:migration.
```

Cette entité est directement liée au SecurityBundle.

```php
<?php

namespace App\Entity;

use App\Repository\UtilisateurRepository;
use Doctrine\ORM\Mapping as ORM;
use Symfony\Component\Security\Core\User\UserInterface;

/**
 * @ORM\Entity(repositoryClass=UtilisateurRepository::class)
 */
class Utilisateur implements UserInterface
{
        /**
         * @ORM\Id()
         * @ORM\GeneratedValue()
         * @ORM\Column(type="integer")
         */
        private $id;

        /**
         * @ORM\Column(type="string", length=180, unique=true)
         */
        private $username;

        /**
         * @ORM\Column(type="json")
```

(suite sur la page suivante)

(suite de la page précédente)

```
     */
    private $roles = [];

    /**
     * @var string The hashed password
     * @ORM\Column(type="string")
     */
    private $password;

    public function getId(): ?int
    {
            return $this->id;
    }

    /**
     * A visual identifier that represents this user.
     *
     * @see UserInterface
     */
    public function getUsername(): string
    {
            return (string) $this->username;
    }

    public function setUsername(string $username): self
    {
            $this->username = $username;

            return $this;
    }

    /**
     * @see UserInterface
     */
    public function getRoles(): array
    {
            $roles = $this->roles;
            // guarantee every user at least has ROLE_USER
            $roles[] = 'ROLE_USER';

            return array_unique($roles);
    }

    public function setRoles(array $roles): self
    {
            $this->roles = $roles;

            return $this;
    }
```

(suite sur la page suivante)

(suite de la page précédente)

```
        /**
         * @see UserInterface
         */
        public function getPassword(): string
        {
                return (string) $this->password;
        }

        public function setPassword(string $password): self
        {
                $this->password = $password;

                return $this;
        }

        /**
         * @see UserInterface
         */
        public function getSalt()
        {
                // not needed when using the "bcrypt" algorithm in␣
↪security.yaml
        }

        /**
         * @see UserInterface
         */
        public function eraseCredentials()
        {
                // If you store any temporary, sensitive data on␣
↪the user, clear it here
                // $this->plainPassword = null;
        }
}
```

Un fichier config/packages/security.yaml est créé. Il contient tous les paramètres pour une gestion sécurisée des utilisateurs.

```
security:
        encoders:
                App\Entity\Utilisateur:
                        algorithm: auto

        # https://symfony.com/doc/current/security.html#where-do-
↪users-come-from-user-providers
        providers:
                # used to reload user from session & other features␣
↪(e.g. switch_user)
```

(suite sur la page suivante)

(suite de la page précédente)

```
                app_user_provider:
                        entity:
                                class: App\Entity\Utilisateur
                                property: username
        firewalls:
                dev:
                        pattern: ^/(_(profiler|wdt)|css|images|js)/
                        security: false
                main:
                        anonymous: lazy
                        provider: app_user_provider

                # activate different ways to authenticate
                # https://symfony.com/doc/current/security.html
↪#firewalls-authentication

                # https://symfony.com/doc/current/security/
↪impersonating_user.html
                # switch_user: true

        # Easy way to control access for large sections of your site
        # Note: Only the *first* access control that matches will␣
↪be used
        access_control:
                # - { path: ^/admin, roles: ROLE_ADMIN }
                # - { path: ^/profile, roles: ROLE_USER }
```

Faites ensuite la migration :

```
php bin/console make:migration
php bin/console doctrine:migrations:migrate
```

Nous pouvons maintenant générer des formulaires spécialement prévu pour de l'authentification :

```
php bin/console make:auth
```

```
netrestore-3c0754698637:securitybundle utilisateur$ php bin/console make:auth

 What style of authentication do you want? [Empty authenticator]:
  [0] Empty authenticator
  [1] Login form authenticator
 > 1

 The class name of the authenticator to create (e.g. AppCustomAuthenticator):
 > LogInFormAuthenticator

 Choose a name for the controller class (e.g. SecurityController) [SecurityController]:
 >

 Do you want to generate a '/logout' URL? (yes/no) [yes]:
 >

 created: src/Security/LogInFormAuthenticator.php
 updated: config/packages/security.yaml
 created: src/Controller/SecurityController.php
 created: templates/security/login.html.twig

  Success!

 Next:
 - Customize your new authenticator.
 - Finish the redirect "TODO" in the App\Security\LogInFormAuthenticator::onAuthenticationSuccess() method.
 - Review & adapt the login template: templates/security/login.html.twig.
```

Cette commande :

— modifie security.yaml
— ajoute un controller
— ajoute une classe héritiaire de AbstractFormLoginAuthenticator
— ajoute un template

Nous avons dès lors un formulaire de login sécurisé ainsi qu'une route pour se déconnecter. Cependant le bundle ne prévoit pas la gestion des utilisateurs. Nous allons donc utiliser le maker bundle pour faire un CRUD.

3.42.4 CRUD utilisateur

```
php bin/console make:crud
```

Comme pour l'authentication à la main, nous allons devoir apporter quelques modifications.

Le controller :

Ajout de use Symfony\Component\Security\Core\Encoder\UserPasswordEncoderInterface ;

Ceci afin de pour crypter le mot de passe saisi lors de l'inscription.

```
<?php
/**
 * @Route("/new", name="utilisateur_new", methods={"GET","POST"})
 */
public function new(Request $request, UserPasswordEncoderInterface
 ↪$passwordEncoder): Response
{
        $utilisateur = new Utilisateur();
```

(suite sur la page suivante)

(suite de la page précédente)

```
        $form = $this->createForm(UtilisateurType::class,
↪$utilisateur);
        $form->handleRequest($request);

        if ($form->isSubmitted() && $form->isValid()) {
                $entityManager = $this->getDoctrine()->getManager();
                //encodage du mot de passe
                $utilisateur->setPassword(
                $passwordEncoder->encodePassword($utilisateur,
↪$utilisateur->getPassword()));
                $entityManager->persist($utilisateur);
                $entityManager->flush();

                return $this->redirectToRoute('utilisateur_index');
        }

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

Le formulaire :

```
<?php

namespace App\Form;

use App\Entity\Utilisateur;
use Symfony\Component\Form\AbstractType;
use Symfony\Component\Form\FormBuilderInterface;
use Symfony\Component\OptionsResolver\OptionsResolver;
//ajout du use pour utiliser le type input password de Symfony
use Symfony\Component\Form\Extension\Core\Type\PasswordType;

class UtilisateurType extends AbstractType
{
        public function buildForm(FormBuilderInterface $builder,␣
↪array $options)
        {
                $builder
                        ->add('username')
                        // suppression du role qui sera défini par␣
↪défaut
                        ->add('password', PasswordType::class)
                ;
        }
```

(suite sur la page suivante)

(suite de la page précédente)

```
        public function configureOptions(OptionsResolver $resolver)
        {
                $resolver->setDefaults([
                'data_class' => Utilisateur::class,
                ]);
        }
}
```

Pour avoir le champ password caché.

Nous modifions maintenant le template du new afin d'ajouter l'input de vérification du mot de passe :

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

{% block title %}New Utilisateur{% endblock %}

{% block body %}
<h1>Create new Utilisateur</h1>

{{ form_start(form, {'attr': {'id': 'new_edit_utilisateur'}}) }}
{# utilisation de classes bootstrap pour la mise en forme #}
        <div class="row">
                <div class="col-12">
                        {{ form_label(form.username) }}
                        {{ form_widget(form.username) }}
                </div>
                <div class="col-12">
                        {{ form_label(form.password) }}
                        {{ form_widget(form.password) }}
                </div>
                <div class="col-12">
                        <label for="verifpass">Saisir une seconde␣
↪fois le mot de passe</label>
                        <input type="password" id="verifpass"␣
↪required>
                </div>
        </div>
        <button class="btn btn-success">{{ button_label|default(
↪'Save') }}</button>
{{ form_end(form) }}

<a href="{{ path('utilisateur_index') }}">back to list</a>
{% endblock %}
```

Création d'un script JS avec jQuery afin de tester si les deux mot de passe sont identiques. Créez tout d'abord un dossier js dans public puis dans ce dossier, un fichier script.js.

Script de vérification :

```
$("#new_edit_utilisateur").on('submit', function(){
    if($("#utilisateur_password").val() != $("#verifpass").val()) {
        //implémntez votre code
        alert("Les deux mots de passe saisies sont différents");
        alert("Merci de renouveler l'opération");
        return false;
    }
})
```

Il ne faut pas oublier d'intégrer le lien vers jQuery et notre script.js dans le base template :

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>{% block title %}Welcome!{% endblock %}</title>
        {% block stylesheets %}
            <link rel="stylesheet" href="https://bootswatch.com/4/
↪yeti/bootstrap.min.css">
        {% endblock %}
    </head>
    <body>
        {% block body %}{% endblock %}
        {% block javascripts %}
            <script
                src="https://code.jquery.com/jquery-3.5.1.min.js"
                integrity="sha256-9/aliU8dGd2tb6OSsuzixeV4y/
↪faTqgFtohetphbbj0="
                crossorigin="anonymous">
            </script>
            <script src="/js/script.js"></script>
        {% endblock %}
    </body>
</html>
```

LoginFormAuthenticator :

Dans le dossier Security nous allons ajouter une route lorsque la connexion a été réalisée avec succès :

```
<?php

public function onAuthenticationSuccess(Request $request,␣
↪TokenInterface $token, $providerKey)
{
        if ($targetPath = $this->getTargetPath($request->
↪getSession(), $providerKey)) {
```

(suite sur la page suivante)

(suite de la page précédente)

```
                return new RedirectResponse($targetPath);
        }
        //on renvoie à la liste des utilisateurs
        return new RedirectResponse($this->urlGenerator->generate(
↪'utilisateur_index'));
}
```

Nous sommes maintenant capable de créer un utilisateur et de se connecter de manière sécurisée. Nous allons maintenant faire un peu de mise en forme. Dans le base template, ajout d'une barre de menu qui sera notre interface de connexion.

Base template :

```
<!DOCTYPE html>
<html>
        <head>
                <meta charset="UTF-8">
                <title>{% block title %}Welcome!{% endblock %}</
↪title>
                {% block stylesheets %}
                        <link rel="stylesheet" href="https://
↪bootswatch.com/4/yeti/bootstrap.min.css">
                {% endblock %}
        </head>
        <body>
                <nav class="navbar navbar-light bg-light">
                        <a class="navbar-brand">Navbar</a>
                        {% if app.user %}
                                <div>
                                        Bonjour {{ app.user.
↪username }} <a class="btn btn-sm btn-danger" href="{{ path('app_
↪logout') }}">Déconnexion</a>
                                </div>
                        {% else %}
                                <div>
                                        <a class="btn btn-sm btn-
↪primary" href="{{ path('utilisateur_new') }}">S'inscrire</a>
                                        <a class="btn btn-sm btn-
↪success" href="{{ path('app_login') }}">Se connecter</a>
                                </div>
                        {% endif %}
                </nav>
                <div class="container">
                        {% if message is defined %}
                                <div class="alert alert-danger">
                                        {{ message }}
                                </div>
                        {% endif %}
```

(suite sur la page suivante)

(suite de la page précédente)

```
                        {% block body %}
                        {% endblock %}
                </div>
                {% block javascripts %}
                        <script
                                src="https://code.jquery.com/jquery-
↪3.5.1.min.js"
                                integrity="sha256-9/
↪aliU8dGd2tb6OSsuzixeV4y/faTqgFtohetphbbj0="
                        crossorigin="anonymous">
                        </script>
                        <script src="/js/script.js"></script>
                {% endblock %}
        </body>
</html>
```

Nous proposons à nos utilisateurs d'acceder aux pages de connexion ou d'inscription s'ils ne sont pas identifiés. Dans le cas contraire nous leur proposons de se déconnecter.

Modification d'un utilisateur

Nous avons besoin lorsque nous allons modifier un utilisateur d'avoir la même double vérification pour le mot de passe.

Dans le template :

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

{% block title %}Edit Utilisateur{% endblock %}

{% block body %}
        <h1>Edit Utilisateur</h1>

        {{ form_start(form, {'attr': {'id': 'new_edit_utilisateur'}}
↪) }}
        {# utilisation de classes bootstrap pour la mise en forme #}
                <div class="row">
                        <div class="col-12">
                                {{ form_label(form.username) }}
                                {{ form_widget(form.username) }}
                        </div>
                        <div class="col-12">
                                {{ form_label(form.password) }}
                                {{ form_widget(form.password) }}
                        </div>
                        <div class="col-12">
                                <label for="verifpass">Saisir une␣
↪seconde fois le mot de passe</label>
```

(suite sur la page suivante)

(suite de la page précédente)

```
                                <input type="password" id="verifpass
↪" required>
                        </div>
                </div>
                <button class="btn btn-success">{{ button_
↪label|default('Update') }}</button>
        {{ form_end(form) }}

        <a href="{{ path('utilisateur_index') }}">back to list</a>

        {{ include('utilisateur/_delete_form.html.twig') }}
{% endblock %}
```

Dans le controller :

```
<?php
/**
 * @Route("/{id}/edit", name="utilisateur_edit", methods={"GET",
↪"POST"})
 */
public function edit(Request $request, Utilisateur $utilisateur,␣
↪UserPasswordEncoderInterface $passwordEncoder): Response
{
        $form = $this->createForm(UtilisateurType::class,
↪$utilisateur);
        $form->handleRequest($request);

        if ($form->isSubmitted() && $form->isValid()) {
                $utilisateur->setPassword($passwordEncoder->
↪encodePassword($utilisateur, $utilisateur->getPassword()));
                $this->getDoctrine()->getManager()->flush();

                return $this->redirectToRoute('utilisateur_index');
        }

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

Nous pouvons maintenant modifier le mot de passe de façon sécurisée.

3.42.5 Navigation avec les droits

Dans cette partie nous allons voir comment autoriser l'accès des pages de notre appliction en fonction des droits (rôles) des utilisateurs.

Note : SecurityBundle gère l'appelle à la route de login si jamais la navigation vers une page requière d'avoir des droits.

Name	No auth	Auth	Admin	Path
page d'accueil	OUI	OUI	OUI	/
inscription	OUI	NON	NON	/utilisateur/new
connexion	OUI	NON	NON	/login
déconnexion	NON	OUI	OUI	/logout
modification	NON	OUI	NON	/utilisateur/{id}/edit
suppression	NON	OUI	OUI	/utilisateur/{id}
page membre	NON	OUI	OUI	/membre
page admin	NON	NON	OUI	/admin

No auth : utilisateur non authentifié Auth : utilisateur authentifié Admin : utilisateur authentifié avec les droits administrateur

Il existe plusieurs manières de procéder. Mais avant toute choses, il faut créer les routes et les templates des nouvelles pages.

Création d'un nouveau controller :

```
php bin/console make:controller
```

Nous l'appelons NavigationController.

```
<?php

namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\Routing\Annotation\Route;

class NavigationController extends AbstractController
{
        /**
         * @Route("/", name="home")
         */
        public function home()
        {
                return $this->render('navigation/home.html.twig');
        }

        /**
         * @Route("/membre", name="membre")
         */
        public function membre()
```

(suite sur la page suivante)

(suite de la page précédente)

```
        {
                return $this->render('navigation/membre.html.twig');
        }

        /**
         * @Route("/admin", name="admin")
         */
        public function admin()
        {
                return $this->render('navigation/admin.html.twig');
        }

}
```

N'oubliez pas de créer les templates allant avec ces nouvelles routes.

Modification du template de base :

```
<!DOCTYPE html>
<html>
        <head>
                <meta charset="UTF-8">
                <title>{% block title %}Welcome!{% endblock %}</
↪title>
                {% block stylesheets %}
                        <link rel="stylesheet" href="https://
↪bootswatch.com/4/yeti/bootstrap.min.css">
                {% endblock %}
        </head>
        <body>
                <nav class="navbar navbar-light bg-light">
                        <a class="navbar-brand">Navbar</a>
                        {% if app.user %}
                                {# Si l'utilisateur est authentifié
↪#}
                                        <a class="btn btn-outline-
↪info" href="{{ path('membre') }}">Page membre</a>
                                {% if is_granted('ROLE_ADMIN') %}
                                        {# Si l'utilisateur est␣
↪authentifié et qu'il est admin #}
                                        <a class="btn btn-outline-
↪warning" href="{{ path('admin') }}">Page admin</a>
                                {% endif %}
                                <div>
                                        Bonjour {{ app.user.
↪username }} <a class="btn btn-sm btn-danger" href="{{ path('app_
↪logout') }}">Déconnexion</a>
                                </div>
```

(suite sur la page suivante)

(suite de la page précédente)

```
                        {% else %}
                                <div>
                                        <a class="btn btn-sm btn-
↪primary" href="{{ path('utilisateur_new') }}">S'inscrire</a>
                                        <a class="btn btn-sm btn-
↪success" href="{{ path('app_login') }}">Se connecter</a>
                                </div>
                        {% endif %}
                </nav>
                <div class="container">
                        {% if message is defined %}
                                <div class="alert alert-danger">
                                        {{ message }}
                                </div>
                        {% endif %}
                        {% block body %}
                        {% endblock %}
                </div>
                {% block javascripts %}
                        <script
                                src="https://code.jquery.com/jquery-
↪3.5.1.min.js"
                                integrity="sha256-9/
↪aliU8dGd2tb6OSsuzixeV4y/faTqgFtohetphbbj0="
                        crossorigin="anonymous">
                        </script>
                        <script src="/js/script.js"></script>
                {% endblock %}
        </body>
</html>
```

Navigation dans le template

Note : Nous l'avons déjà vu dans le base template précédemment.

Utilisation de {% if is_granted("<<Droit à tester>>") %} et de {% if app.user %} pour savoir si il y a une connexion en cours.

Navigation dans le controller

Utilisation des annotations :

Il est possible d'annoter chaque méthode.

```php
<?php

namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\Routing\Annotation\Route;
// Le use pour les annotations de sécurité
use Sensio\Bundle\FrameworkExtraBundle\Configuration\IsGranted;

class NavigationController extends AbstractController
{
        /**
         * @Route("/", name="home")
         */
        public function home()
        {
                return $this->render('navigation/home.html.twig');
        }

        /**
         * Nécessite juste d'être connecté
         * @Route("/membre", name="membre")
         * @IsGranted("IS_AUTHENTICATED_FULLY")
         * fonctionne aussi avec ROLE_USER
        */
        public function membre()
        {
                return $this->render('navigation/membre.html.twig');
        }

        /**
         * Besoin des droits admin
         * @Route("/admin", name="admin")
         * @IsGranted("ROLE_ADMIN")
        */
        public function admin()
        {
                return $this->render('navigation/admin.html.twig');
        }

}
```

Ainsi que pour tout le controller (dans ce cas toutes les routes ne seront accessibles que si les droits requis sont respéctés) :

```php
<?php

namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
```

(suite sur la page suivante)

(suite de la page précédente)

```
use Symfony\Component\Routing\Annotation\Route;
use Sensio\Bundle\FrameworkExtraBundle\Configuration\IsGranted;

/**
* @IsGranted("ROLE_ADMIN")
*/
class NavigationController extends AbstractController
{
        /**
         * @Route("/", name="home")
         */
        public function home()
        {
                return $this->render('navigation/home.html.twig');
        }

        /**
         * @Route("/membre", name="membre")
         */
        public function membre()
        {
                return $this->render('navigation/membre.html.twig');
        }

        /**
         * @Route("/admin", name="admin")
         */
        public function admin()
        {
                return $this->render('navigation/admin.html.twig');
        }

}
```

Vérification dans la méthode :

```
<?php
/**
 * @Route("/membre", name="membre")
 */
public function membre()
{
        //test si un utilisateur est connecté
        $this->denyAccessUnlessGranted('IS_AUTHENTICATED_FULLY');
        return $this->render('navigation/membre.html.twig');
}
```

Avertissement : Ces méthodes ne gèrent pas les redirections en cas de refus d'accès (ex : droit admin). Il est fortement recommander de n'utiliser ces méthodes que pour bloquer un utilisateur non connecté. IS_AUTHENTICATED_FULLY.

Pour terminer avec le controller, il est possible de faire une vérification manuelle :

```
<?php
/**
 * @Route("/admin", name="admin")
 */
public function admin(Session $session)
{
        //récupération de l'utilisateur security>Bundle
        $utilisateur = $this->getUser();

        //vérification des droits.
        if($utilisateur && in_array('ROLE_ADMIN', $utilisateur->
↪getRoles())){
                return $this->render('navigation/admin.html.twig');
        }

        //redirection
        $session->set("message", "Vous n'avez pas le droit d
↪'acceder à la page admin vous avez été redirigé sur cette page");
        return $this->redirectToRoute('home');

}
```

Note : Cette méthode est fortement recommandée lorqu'il faut gérer des accès par droits spéciaux.

Navigation dans la configuration

Dans le fichier config/packages/security.yaml

```
security:
        [... code existant]
        # Easy way to control access for large sections of your site
        # Note: Only the *first* access control that matches will␣
↪be used
        access_control:
                - { path: ^/admin, roles: ROLE_ADMIN }
                - { path: ^/membre, roles: ROLE_USER }
               # ou -  { path: ^/membre, roles: IS_AUTHENTICATED_
↪FULLY }
```

Dans la sous partie access_control de security, il suffit de saisir les URL suivie des droits d'accès.

3.43 Exemple final

Maintenant je vais vous présenter la solution que j'ai retenu pour mon exemple. En fait il s'agit d'un mix de toutes les méthodes vues précédemment.

3.43.1 Fichier de config

security.yaml

```
security:
        encoders:
                App\Entity\Utilisateur:
                        algorithm: auto

        # https://symfony.com/doc/current/security.html#where-do-
↪users-come-from-user-providers
        providers:
                # used to reload user from session & other features␣
↪(e.g. switch_user)
                app_user_provider:
                        entity:
                                class: App\Entity\Utilisateur
                                property: username
        firewalls:
                dev:
                        pattern: ^/(_(profiler|wdt)|css|images|js)/
                        security: false
                main:
                        anonymous: lazy
                        provider: app_user_provider
                        guard:
                                authenticators:
                                -␣
↪App\Security\LogInFormAuthenticator
                        logout:
                                path: app_logout
                                # where to redirect after logout
                                target: home

                # activate different ways to authenticate
                # https://symfony.com/doc/current/security.html
↪#firewalls-authentication
```

(suite sur la page suivante)

(suite de la page précédente)

```
                # https://symfony.com/doc/current/security/
↪impersonating_user.html
                # switch_user: true

        # Easy way to control access for large sections of your site
        # Note: Only the *first* access control that matches will␣
↪be used
        access_control:
                - { path: ^/logout, roles: ROLE_USER }

                # permet de rendre la route /new accessible pour␣
↪les utilisateurs anonymes (non connecté)
                - { path: ^/utilisateur/new, roles: IS_
↪AUTHENTICATED_ANONYMOUSLY }

                # bloque toutes les routes commençant par /
↪utilisateur sauf la ligne du dessus
                - { path: ^/utilisateur, roles: ROLE_USER }
                - { path: ^/membre, roles: IS_AUTHENTICATED_FULLY }
```

Je ne l'utilise que pour les routes nécessitant d'être connecté (sans distinction de droits).

Note : ROLE_USER == IS_AUTHENTICATED_FULLY

3.43.2 Controllers

NavigationController

```
<?php

namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\Routing\Annotation\Route;
use Symfony\Component\HttpFoundation\Session\Session;

class NavigationController extends AbstractController
{
        /**
         * @Route("/", name="home")
         */
        public function home(Session $session)
        {
                $return = [];
```

(suite sur la page suivante)

(suite de la page précédente)

```
                if($session->has('message'))
                {
                        $message = $session->get('message');
                        $session->remove('message'); //on vide la␣
↪variable message dans la session
                        $return['message'] = $message; //on ajoute␣
↪à l'array de paramètres notre message
                }
                return $this->render('navigation/home.html.twig',
↪$return);
        }

        /**
         * @Route("/membre", name="membre")
         */
        public function membre(Session $session)
        {
                $return = [];

                if($session->has('message'))
                {
                        $message = $session->get('message');
                        $session->remove('message'); //on vide la␣
↪variable message dans la session
                        $return['message'] = $message; //on ajoute␣
↪à l'array de paramètres notre message
                }
                return $this->render('navigation/membre.html.twig',
↪$return);
        }

        /**
         * @Route("/admin", name="admin")
         */
        public function admin(Session $session)
        {
                $utilisateur = $this->getUser();
                if(!$utilisateur)
                {
                        $session->set("message", "Merci de vous␣
↪connecter");
                        return $this->redirectToRoute('app_login');
                }

                else if(in_array('ROLE_ADMIN', $utilisateur->
↪getRoles())){
                        return $this->render('navigation/admin.html.
↪twig');
                }
```

(suite sur la page suivante)

(suite de la page précédente)

```
                $session->set("message", "Vous n'avez pas le droit d
↪'acceder à la page admin vous avez été redirigé sur cette page");
                if($session->has('message'))
                {
                        $message = $session->get('message');
                        $session->remove('message'); //on vide la␣
↪variable message dans la session
                        $return['message'] = $message; //on ajoute␣
↪à l'array de paramètres notre message
                }
                return $this->redirectToRoute('home', $return);

        }

}
```

UtilisateurController

```
<?php

namespace App\Controller;

use App\Entity\Utilisateur;
use App\Form\UtilisateurType;
use App\Repository\UtilisateurRepository;
use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\Request;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Routing\Annotation\Route;
use␣
↪Symfony\Component\Security\Core\Encoder\UserPasswordEncoderInterface;
↪
use Symfony\Component\HttpFoundation\Session\Session;

/**
* @Route("/utilisateur")
*/
class UtilisateurController extends AbstractController
{
        /**
        * @Route("/", name="utilisateur_index", methods={"GET"})
        */
        public function index(UtilisateurRepository
↪$utilisateurRepository, Session $session): Response
        {
                //besoin de droits admin
                $utilisateur = $this->getUser();
```

(suite sur la page suivante)

(suite de la page précédente)

```
                if(!$utilisateur)
                {
                        $session->set("message", "Merci de vous␣
↪connecter");
                        return $this->redirectToRoute('app_login');
                }

                else if(in_array('ROLE_ADMIN', $utilisateur->
↪getRoles())){
                        return $this->render('utilisateur/index.
↪html.twig', [
                                'utilisateurs' =>
↪$utilisateurRepository->findAll(),
                        ]);
                }

                return $this->redirectToRoute('home');
        }

        /**
         * @Route("/new", name="utilisateur_new", methods={"GET",
↪"POST"})
         */
        public function new(Request $request,␣
↪UserPasswordEncoderInterface $passwordEncoder, Session $session):␣
↪Response
        {

                //test de sécurité, un utilisateur connecté ne peut␣
↪pas s'inscrire
                $utilisateur = $this->getUser();
                if($utilisateur)
                {
                        $session->set("message", "Vous ne pouvez␣
↪pas créer un compte lorsque vous êtes connecté");
                        return $this->redirectToRoute('membre');
                }

                $utilisateur = new Utilisateur();
                $form = $this->createForm(UtilisateurType::class,
↪$utilisateur);
                $form->handleRequest($request);

                if ($form->isSubmitted() && $form->isValid()) {
                        $entityManager = $this->getDoctrine()->
↪getManager();
                        $utilisateur->setPassword($passwordEncoder->
↪encodePassword($utilisateur, $utilisateur->getPassword()));
                        /* uniquement pour créer un admin
```

(suite sur la page suivante)

(suite de la page précédente)

```
                        $role = ['ROLE_ADMIN'];
                        $utilisateur->setRoles($role); */
                        $entityManager->persist($utilisateur);
                        $entityManager->flush();

                        return $this->redirectToRoute('utilisateur_
↪index');
                }

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

        /**
         * @Route("/{id}", name="utilisateur_show", methods={"GET"})
         */
        public function show(Utilisateur $utilisateur): Response
        {
                //accès géré dans le security.yaml
                return $this->render('utilisateur/show.html.twig', [
                'utilisateur' => $utilisateur,
                ]);
        }

        /**
         * @Route("/{id}/edit", name="utilisateur_edit", methods={
↪"GET","POST"})
         */
        public function edit(Request $request, Utilisateur
↪$utilisateur, UserPasswordEncoderInterface $passwordEncoder,␣
↪Session $session, $id): Response
        {
                $utilisateur = $this->getUser();
                if($utilisateur->getId() != $id )
                {
                        // un utilisateur ne peut pas en modifier␣
↪un autre
                        $session->set("message", "Vous ne pouvez␣
↪pas modifier cet utilisateur");
                        return $this->redirectToRoute('membre');
                }
                $form = $this->createForm(UtilisateurType::class,
↪$utilisateur);
                $form->handleRequest($request);

                if ($form->isSubmitted() && $form->isValid()) {
                        $utilisateur->setPassword($passwordEncoder->
↪encodePassword($utilisateur, $utilisateur->getPassword()));
```

(suite sur la page suivante)

(suite de la page précédente)

```
                        $this->getDoctrine()->getManager()->flush();

                        return $this->redirectToRoute('utilisateur_
↪index');
                }

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

        /**
         * @Route("/{id}", name="utilisateur_delete", methods={
↪"DELETE"})
         */
        public function delete(Request $request, Utilisateur
↪$utilisateur, Session $session, $id): Response
        {
                $utilisateur = $this->getUser();
                if($utilisateur->getId() != $id )
                {
                        // un utilisateur ne peut pas en supprimer␣
↪un autre
                        $session->set("message", "Vous ne pouvez␣
↪pas supprimer cet utilisateur");
                        return $this->redirectToRoute('membre');
                }

                if ($this->isCsrfTokenValid('delete'.$utilisateur->
↪getId(), $request->request->get('_token')))
                {
                        $entityManager = $this->getDoctrine()->
↪getManager();
                        $entityManager->remove($utilisateur);
                        $entityManager->flush();
                        // permet de fermer la session utilisateur␣
↪et d'éviter que l'EntityProvider ne trouve pas la session
                        $session = new Session();
                        $session->invalidate();
                }

                return $this->redirectToRoute('home');
        }
}
```

SecurityController

```
<?php

namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Routing\Annotation\Route;
use␣
↪Symfony\Component\Security\Http\Authentication\AuthenticationUtils;
↪
use Symfony\Component\HttpFoundation\Session\Session;

class SecurityController extends AbstractController
{
        /**
         * @Route("/login", name="app_login")
         */
        public function login(AuthenticationUtils
↪$authenticationUtils, Session $session): Response
        {
                // if ($this->getUser()) {
                //     return $this->redirectToRoute('target_path');
                // }

                // get the login error if there is one
                $error = $authenticationUtils->
↪getLastAuthenticationError();
                // last username entered by the user
                $lastUsername = $authenticationUtils->
↪getLastUsername();

                $return = ['last_username' => $lastUsername, 'error
↪' => $error];

                if($session->has('message'))
                {
                        $message = $session->get('message');
                        $session->remove('message'); //on vide la␣
↪variable message dans la session
                        $return['message'] = $message; //on ajoute␣
↪à l'array de paramètres notre message
                }

                return $this->render('security/login.html.twig',
↪$return);
        }

        /**
```

(suite sur la page suivante)

(suite de la page précédente)

```
         * @Route("/logout", name="app_logout")
         */
        public function logout()
        {
                throw new \LogicException('This method can be blank␣
↪- it will be intercepted by the logout key on your firewall.');
        }
}
```

3.43.3 Templates

base.html.twig

```
<!DOCTYPE html>
<html>
        <head>
                <meta charset="UTF-8">
                <title>{% block title %}Welcome!{% endblock %}</
↪title>
                {% block stylesheets %}
                <link rel="stylesheet" href="https://bootswatch.com/
↪4/yeti/bootstrap.min.css">
                <link rel="stylesheet" href="/css/style.css">
                {% endblock %}
        </head>
        <body>
                <nav class="navbar navbar-light bg-light">
                        <a class="navbar-brand" href="{{ path('home
↪') }}">Accueil</a>
                        {% if app.user %}
                                <a class="btn btn-outline-info"␣
↪href="{{ path('membre') }}">Page membre</a>
                                {% if is_granted('ROLE_ADMIN') %}
                                        <a class="btn btn-outline-
↪warning" href="{{ path('admin') }}">Page admin</a>
                                {% endif %}
                                <div>
                                        Bonjour {{ app.user.
↪username }} <a class="btn btn-sm btn-danger" href="{{ path('app_
↪logout') }}">Déconnexion</a>
                                </div>
                        {% else %}
                                <div>
                                        <a class="btn btn-sm btn-
↪primary" href="{{ path('utilisateur_new') }}">S'inscrire</a>
                                        <a class="btn btn-sm btn-
↪success" href="{{ path('app_login') }}">Se connecter</a>
```

(suite sur la page suivante)

(suite de la page précédente)

```
                                </div>
                        {% endif %}
                </nav>
                <div class="container">
                        {% if message is defined %}
                                <div class="alert alert-danger">
                                        {{ message }}
                                </div>
                        {% endif %}
                        {% block body %}
                        {% endblock %}
                </div>
                <footer class="navbar-light bg-light">
                        <div class="footer-copyright text-center py-
↪3">© 2020 Copyright:
                                <a href="https://www.univ-orleans.
↪fr/iut-orleans/informatique/intra/tuto/php/"> Gérard Rozsavolgyi &
↪ Sylvain Austruy</a>
                        </div>
                </footer>
                        {% block javascripts %}
                        <script
                                src="https://code.jquery.com/jquery-
↪3.5.1.min.js"
                                integrity="sha256-9/
↪aliU8dGd2tb6OSsuzixeV4y/faTqgFtohetphbbj0="
                        crossorigin="anonymous">
                        </script>
                        <script src="/js/script.js"></script>
                {% endblock %}
        </body>
</html>
```

new.html.twig

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

{% block title %}Inscription{% endblock %}

{% block body %}
        <h1>Inscription</h1>

        <fieldset>
                <legend>Inscrivez vous</legend>
                {{ form_start(form, {'attr': {'id': 'new_edit_
↪utilisateur'}}) }}
                {# utilisation de classes bootstrap pour la mise en␣
↪forme #}
```

(suite sur la page suivante)

(suite de la page précédente)

```
                <div class="row">
                        <div class="col-12">
                                {{ form_label(form.username) }}
                                {{ form_widget(form.username) }}
                                {{ form_errors(form.username) }}
                        </div>
                        <div class="col-12">
                                {{ form_label(form.password) }}
                                {{ form_widget(form.password) }}
                        </div>
                        <div class="col-12">
                                <label for="verifpass">Saisir une␣
↪seconde fois le mot de passe</label>
                                <input type="password" id="verifpass
↪" required>
                        </div>
                </div>
                <button class="btn btn-success">{{ button_
↪label|default('S\'enregistrer') }}</button>
                {{ form_end(form) }}
        </fieldset>

        <a class="btn btn-info" href="{{ path('home') }}">Page d
↪'accueil</a>
{% endblock %}
```

Ajout de {{ form_errors(form.username) }} pour afficher une erreur quand le pseudo est déjà utilisé (la vérification se fait automatiquement, car nous avons choisi l'attribut username comme unique) et un peu de css.

edit.html.twig

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

{% block title %}Modification compte{% endblock %}

{% block body %}
        <h1>Modifier votre compte</h1>
        <fieldset>
                <legend>Modification</legend>
                {{ form_start(form, {'attr': {'id': 'new_edit_
↪utilisateur'}}) }}
                {# utilisation de classes bootstrap pour la mise en␣
↪forme #}
                        <div class="row">
                                <div class="col-12">
                                        {{ form_label(form.
↪username) }}
```

(suite sur la page suivante)

(suite de la page précédente)

```
                                        {{ form_widget(form.
↪username) }}
                                </div>
                                <div class="col-12">
                                        {{ form_label(form.
↪password) }}
                                        {{ form_widget(form.
↪password) }}
                                </div>
                                <div class="col-12">
                                        <label for="verifpass">
↪Saisir une seconde fois le mot de passe</label>
                                        <input type="password" id=
↪"verifpass" required>
                                </div>
                        </div>
                        <button class="btn btn-success">{{ button_
↪label|default('Mettre à jour') }}</button>
                {{ form_end(form) }}
        </fieldset>

        <a class="btn btn-outline-primary" href="{{ path('membre') }
↪}">Page membre</a>

        {{ include('utilisateur/_delete_form.html.twig') }}
{% endblock %}
```

show.html.twig

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

{% block title %}Utilisateur{% endblock %}

{% block body %}
        <h1>Utilisateur</h1>

        <table class="table">
                <tbody>
                <tr>
                                <th>Id</th>
                                <td>{{ utilisateur.id }}</td>
                        </tr>
                        <tr>
                                <th>Username</th>
                                <td>{{ utilisateur.username }}</td>
                        </tr>
                        <tr>
```

(suite sur la page suivante)

(suite de la page précédente)

```
                                <th>Roles</th>
                                <td>{{ utilisateur.roles ?␣
↪utilisateur.roles|json_encode : '' }}</td>
                        </tr>
                        <tr>
                                <th>Password</th>
                                <td>{{ utilisateur.password }}</td>
                        </tr>
                </tbody>
        </table>

        <a class="btn btn-outline-primary mt-2" href="{{ path(
↪'membre') }}">Page membre</a>

        <a class="btn btn-warning mt-2" href="{{ path('utilisateur_
↪edit', {'id': utilisateur.id}) }}">Modifier</a>

        {{ include('utilisateur/_delete_form.html.twig') }}
{% endblock %}
```

login.html.twig

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

{% block title %}Connexion{% endblock %}

{% block body %}
        <form method="post" class="mt-5">
                <fieldset>
                        <legend>Connectez vous</legend>

                        {% if error %}
                                <div class="alert alert-danger">{{␣
↪error.messageKey|trans(error.messageData, 'security') }}</div>
                        {% endif %}

                        {% if app.user %}
                                <div class="mb-3">
                                        Bonjour {{ app.user.
↪username }} <a class="btn btn-sm btn-danger" href="{{ path('app_
↪logout') }}">Déconnexion</a>
                                </div>
                        {% else %}
                                <div class="row">
                                        <div class="col-12 mt-3">
                                                <input type="text"␣
↪value="{{ last_username }}" name="username" id="inputUsername"␣
↪class="form-control" required autofocus placeholder="pseudo">
```

(suite sur la page suivante)

(suite de la page précédente)

```
                                        </div>
                                        <div class="col-12 mt-3 mb-3
↪">
                                                <input type=
↪"password" name="password" id="inputPassword" class="form-control
↪" required placeholder="mot de passe">
                                        </div>
                                </div>
                                <button class="btn btn-success"␣
↪type="submit">
                                        Connexion
                                </button>
                        {% endif %}

                        <input type="hidden" name="_csrf_token"
                        value="{{ csrf_token('authenticate') }}"
                        >

                        {#
                        Uncomment this section and add a remember_
↪me option below your firewall to activate remember me␣
↪functionality.
                        See https://symfony.com/doc/current/
↪security/remember_me.html

                        <div class="checkbox mb-3">
                                <label>
                                        <input type="checkbox" name=
↪"_remember_me"> Remember me
                                </label>
                        </div>
                        #}
                </fieldset>
        </form>
{% endblock %}
```

Les templates de NavigationController

Les templates pour tester la navigation sécurisée.

Admin :

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

{% block title %}Page admin{% endblock %}

{% block body %}
        <h1>Ma page admin</h1>
```

(suite sur la page suivante)

(suite de la page précédente)

```
        <h2>{{ app.user.username }} tu es un admin</h2>

        <p>
                You've probably heard of Lorem Ipsum before - it's␣
↪the most-used dummy text excerpt out there.
                People use it because it has a fairly normal␣
↪distribution of letters and words (making it look like normal␣
↪English),
                but it's also Latin, which means your average␣
↪reader won't get distracted by trying to read it.
                It's perfect for showcasing design work as it␣
↪should look when fleshed out with text, because it allows viewers␣
↪to focus on the design work itself, instead of the text.
                It's also a great way to showcase the functionality␣
↪of programs like word processors, font types, and more.
                We've taken Lorem Ipsum to the next level with our␣
↪HTML-Ipsum tool.
                As you can see, this Lorem Ipsum is tailor-made for␣
↪websites and other online projects that are based in HTML. Most␣
↪web design projects use Lorem Ipsum excerpts to begin with,
                but you always have to spend an annoying extra␣
↪minute or two formatting it properly for the web.
                Maybe you have a custom-styled ordered list you␣
↪want to show off, or maybe you just want a long chunk of Lorem␣
↪Ipsum that's already wrapped in paragraph tags.
                No matter the need, we've put together a number of␣
↪Lorem Ipsum samples ready to go with HTML tags and formatting in␣
↪place.
                All you have to do is click the heading of any␣
↪section on this page, and that HTML-Ipsum block is copied to your␣
↪clipboard
                nd ready to be loaded into a website redesign␣
↪template, brand new design mockup, or any other digital project␣
↪you might need dummy text for.
                No matter the project, please remember to replace␣
↪your fancy HTML-Ipsum with real content before you go live - t
                his is especially important if you're planning to␣
↪implement a content-based marketing strategy for your new␣
↪creation!
                Lorem Ipsum text is all well and good to␣
↪demonstrate a design or project, but if you set a website loose␣
↪on the Internet
                without replacing dummy text with relevant, high␣
↪quality content, you'll run into all sorts of potential Search␣
↪Engine Optimization issues
                like thin content, duplicate content, and more.
                HTML-Ipsum is maintained by WebFX.
                For more information about us, please visit WebFX␣
↪Reviews.
```

(suite sur la page suivante)

(suite de la page précédente)

```
                To learn more about the industries we drive␣
↪Internet marketing performormance for and our revenue driving␣
↪services:
                SEO, PPC, social media, web design, local SEO and␣
↪online advertising services.
        </p>
        <a class="btn btn-primary mt-3" href="{{ path('utilisateur_
↪edit', {'id': app.user.id}) }}">Modifier mon compte</a>
        <a class="btn btn-info mt-3" href="{{ path('utilisateur_
↪index') }}">Voir la liste des utilisateurs inscrit</a>
{% endblock %}
```

Home :

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

{% block title %}Page d'accueil{% endblock %}

{% block body %}
        <h1>Mon super site</h1>

        <p><strong>Pellentesque habitant morbi tristique</strong>␣
↪senectus et netus et malesuada fames ac turpis egestas.␣
↪Vestibulum tortor quam,
        feugiat vitae, ultricies eget, tempor sit amet, ante. Donec␣
↪eu libero sit amet quam egestas semper. <em>Aenean ultricies mi␣
↪vitae est.</em> Mauris
        placerat eleifend leo. Quisque sit amet est et sapien␣
↪ullamcorper pharetra. Vestibulum erat wisi, condimentum sed,
↪<code>commodo vitae</code>, ornare sit amet, wisi.
         Aenean fermentum, elit eget tincidunt condimentum, eros␣
↪ipsum rutrum orci, sagittis tempus lacus enim ac dui. <a href="#">
↪Donec non enim</a> in turpis pulvinar facilisis. Ut felis.</p>

        <h2>Header Level 2</h2>
        <ol>
                <li>Lorem ipsum dolor sit amet, consectetuer␣
↪adipiscing elit.</li>
                <li>Aliquam tincidunt mauris eu risus.</li>
        </ol>
        <blockquote><p>Lorem ipsum dolor sit amet, consectetur␣
↪adipiscing elit. Vivamus magna.
        Cras in mi at felis aliquet congue. Ut a est eget ligula␣
↪molestie gravida. Curabitur massa.
        Donec eleifend, libero at sagittis mollis, tellus est␣
↪malesuada tellus, at luctus turpis elit sit amet quam. Vivamus␣
↪pretium ornare est.</p></blockquote>

        <h3>Header Level 3</h3>
```

(suite sur la page suivante)

(suite de la page précédente)

```
        <ul>
                <li>Lorem ipsum dolor sit amet, consectetuer␣
↪adipiscing elit.</li>
                <li>Aliquam tincidunt mauris eu risus.</li>
        </ul>
        <pre><code>
                #header h1 a {
                display: block;
                width: 300px;
                height: 80px;
                }
        </code></pre>
{% endblock %}
```

Membre :

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

{% block title %}Page membre{% endblock %}

{% block body %}
        <h1>Mon espace membre</h1>

        <h2>{{ app.user.username }} voici ton espace personnel</h2>

        <dl>
                <dt>Definition list</dt>
                <dd>Consectetur adipisicing elit, sed do eiusmod␣
↪tempor incididunt ut labore et dolore magna
                aliqua. Ut enim ad minim veniam, quis nostrud␣
↪exercitation ullamco laboris nisi ut aliquip ex ea
                commodo consequat.</dd>
                <dt>Lorem ipsum dolor sit amet</dt>
                <dd>Consectetur adipisicing elit, sed do eiusmod␣
↪tempor incididunt ut labore et dolore magna
                aliqua. Ut enim ad minim veniam, quis nostrud␣
↪exercitation ullamco laboris nisi ut aliquip ex ea
                commodo consequat.</dd>
        </dl>

        <ol>
                <li>Lorem ipsum dolor sit amet, consectetuer␣
↪adipiscing elit.</li>
                <li>Aliquam tincidunt mauris eu risus.</li>
                <li>Vestibulum auctor dapibus neque.</li>
        </ol>


        <a class="btn btn-primary" href="{{ path('utilisateur_edit',
↪ {'id': app.user.id}) }}">Modifier mon compte</a>
```

(suite sur la page suivante)

(suite de la page précédente)

```
{% endblock %}
```

3.43.4 CSS

public/css/style.css

```
fieldset {
        margin: 0 auto;
        width: 50%;
        padding: 1em;
        border: 1px solid #CCC;
        border-radius: 1em;
}

legend {
        width: 50%;
}

body {
        margin: 0;
        min-height: 100vh;
        display: grid;
        grid-template-rows: auto 1fr auto;
}
```

Note : Ce tuto est terminé, il ne vous reste qu'à l'adapter à votre projet. Ainsi que de réaliser un beau CSS avec Bootstrap ou autre.

3.44 Feuilles de TD Lic Pro Web et Mobile

3.44.1 Année 2018-2019

— TD1
— TD2
— TD3

3.45 Feuilles de TD 2ème Année IUT informatique

3.45.1 Année 2018-2019

— TD1
— TP1
— TD2
— TP2
— TP3
— TD3
— Git init

3.46 Feuilles de TD Lic Pro Web et Mobile

3.46.1 Année 2017-2018

— TD1
— TD2
— TD3
— TD4
— TD5
— TD6
— TD7

3.47 Feuilles de TD CVRH Tours

3.47.1 Formation de Juin 2015

— TD1
— TD2
— TD3

3.48 Alice

3.48.1 Paramétrage et initialisations :

```
alice> git config --global user.name "Alice Torvalds"
alice> git config --global user.email "alice@kernel.org"
alice> git config -l
```

3.48.2 Création d'un dossier local versionné

```
mkdir monprojet
cd monprojet
git init
```

ou directement :

```
git init monprojet
```

3.48.3 Ajout de contenu dans monprojet

```
echo "# Mon projet" > Readme.md
git add Readme.md
```

3.48.4 On commite

```
git commit -m "ajout Readme.md"
```

3.48.5 On vérifie l'historique

```
git log
```

ou

```
git log --oneline
```

3.48.6 Ajout d'un nouveau fichier par Alice

```
echo "# Installation" > install.md
```

qui crée un nouveau fichier install.md

puis git status, add et commit :

```
alice> git status
alice> git add install.md
alice> git commit -m "Ajout install.md"
```

3.48.7 Modification du fichier Readme.md

```
echo "[Installation](install.md)" >> Readme.md
```

on peut faire à nouveau un add de Readme.md puis un commit ou plus simplement :

```
git commit -am "Modif Readme.md"
```

qui réalise l'ajout et le commit simultanément.

3.48.8 git status et git show

Si vous voulez savoir où vous en êtes, n'oubliez jamais la commande magique :

```
alice> git status
```

et pour avoir des détails sur la contribution d'un commit, faire un `git show` sur son id (sha1, prononcez **chaone**, les premiers chiffres suffisent), par exemple :

```
git show d2ee25
```

3.48.9 Alice crée un dépot (privé) "monprojet" sur Gitlab

Les dépôts sur Gitlab peuvent être :

- privés
- publics
- ou internes à gitlab

On peut aussi en créer sur :

Bitbucket (https ://bitbucket.org/)

ou

Github (https ://github.com/)

3.48.10 Connexion entre le local et le gitlab

```
alice> git remote add origin https://gitlab.com/alice/monprojet.git
```

3.48.11 Alice pousse son master sur son remote

```
alice> git push -u origin master
```

puis ultérieurement :

```
alice> git push
```

```
alice> git push --follow-tags
```

pour envoyer aussi les tags concernés dans le push

3.48.12 Autres réglages git

Mémoriser les identifiants

Sous Linux

```
alice> git config --global credential.helper 'cache --timeout=7200'
```

mémorise vos identifiants pendant 2h

Sous Windows, *gitbash* est configuré par défaut avec le manager

```
alice> git config --global credential.helper manager
```

le manager gère dans ce cas les identifiants.

Configurer son éditeur favori

```
alice> git config --global core.editor emacs
```

ou

```
alice> git config --global core.editor "code --wait"
```

Fichier .gitconfig

Il se trouve généralement dans le HOME de l'utilisateur, à la racine, une configuration typique avec vscode :

```
[user]
    name = Linus Torvalds
    email = linus@git.edu
[diff]
    tool = default-difftool
[difftool "default-difftool"]
    cmd = code --wait --diff $LOCAL $REMOTE
[push]
```

(suite sur la page suivante)

(suite de la page précédente)

```
    default = simple
   followTags = true
[core]
    editor = code --wait
[color]
    ui = auto
[credential]
    helper = cache --timeout=7200
[merge]
    tool = vscode
[mergetool "vscode"]
    cmd = code --wait $MERGED
```

3.48.13 Bob collabore avec Alice

Fork

Bob va faire un *fork* du dépôt d'Alice sur Gitlab, puis cloner son propre fork pour obtenir une copie locale : `bash git clone https://gitlab.com/bob/monprojet.git`

remotes

— Bob possède déjà un remote : Le sien, *origin*. Il peut le vérifier avec la commande `git remote -v`
— il peut ajouter celui d'Alice :

```
bob> git remote add alice https://gitlab.com/alice/monprojet.git
```

— puis vérifier ses 2 remotes en tapant `git remote -v`

Bob pourra maintenant collaborer avec Alice.

3.48.14 Réalisation d'une fonctionnalité par Alice :

— Alice prend une chose à réaliser et implémente le code nécessaire
— Alice fait les tests et vérifie que ça marche
— git commit -am « Message de commit »

3.48.15 Alice pousse son master sur son remote :

```
git push -u origin master
```

3.49 Bob travaille en collaboration avec Alice grâce à git :

Bob fait d'abord comme Alice pour paramétrer et initialiser son dépôt local. Bob et Alice vont travailler tout d'abord sur leur branche master. Cela peut sembler plus simple mais en fait on va passer rapidement à un travail dans des branches spécifiques à chaque tâche : *GIT branches*

3.49.1 Bob configure ses remotes :

Bob doit avoir 2 remotes :

— Le sien, origin qu'il crée au besoin en faisant :

```
git remote add origin https://gitlab.com/bob/monprojet.git
```

— celui d'Alice qu'il ajoute :

```
git remote add alice https://gitlab.com/alice/monprojet.
↪git
```

— il tape git remote -v pour vérifier ses remotes

— **Si il se trompe :**

```
git remote remove alice
```

3.49.2 Bob récupère le master d'Alice :

```
git fetch Alice master
```

3.49.3 Bob consulte la branche locale correspondant au master d'Alice :

```
git branch -av
git checkout Alice/master
```

puis vérifie que le code d'Alice est correct

3.49.4 Bob revient dans son master :

```
git checkout master
```

3.49.5 Bob merge le travail d'Alice et pousse les modifs dans son dépôt distant :

```
git merge Alice/master
git push
```

Puis detruit la branche locale d'Alice :

```
git branch -d Alice/master
```

3.50 Alice se met à jour :

— ajoute le remote de Bob
— **fetche le master de Bob pour se mettre à jour :**

```
git fetch Bob master
```

— **Fusionne :**

```
git merge Bob/master
```

3.51 Corriger des erreurs Git

3.51.1 Défaire le dernier commit

Plusieurs solutions possibles mais la plus simple est :

```
git reset HEAD^
```

seul le commit est enlevé, le code est toujours présent.

3.51.2 Supprimer une révision

```
git revert 32ee587
```

supprime la révision indiquée de l'historique tout en créant un nouveau commit inversant les changements effectués auparavant

3.52 Scénario de travail collaboratif à l'aide de branches

3.52.1 le problème

Supposons qu'Alice et Bob veulent collaborer sur le projet *super-proj* - Alice possède le code initial qu'elle publie sur son dépôt gitlab : https://gitlab.com/alice/super-proj.git - Bob fait un *fork* du projet d'Alice sur Gitlab - il obtient ainsi sa propre copie de *super-proj* à l'URL : https://gitlab.com/bob/super-proj.git, qu'il peut cloner pour en obtenir une copie locale :

```
git clone https://gitlab.com/bob/super-proj.git
```

— Alice invite Bob sur son projet en tant que *Reporter*
— Bob invite Alice sur son projet en tant que *Reporter*
— Alice ajoute le remote de Bob :

```
git remote add bob https://gitlab.com/bob/super-proj.git
```

— Bob ajoute le remote d'Alice :

```
git remote add alice https://gitlab.com/alice/super-proj.git
```

— chacun vérifie qu'il a 2 remotes : origin et celui de son collaborateur en tapant :

```
git remote -v
```

(qui doit afficher 4 lignes)

Le travail collaboratif

Bob

3.52.2 Création d'une branche bd et utilisation de celle-ci :

```
git branch bd
git checkout bd
```

ou en une seule ligne :

```
git checkout -b bd
```

— .... Fait le travail/Teste puis commite :

```
git commit -am "Intégration BD"
```

3.52.3 Bob pousse sa branche sur son remote :

```
git push origin bd
```

Bob se rend sur le dépôt d'Alice et y fait un *Merge Request* (Gitlab) ou *Pull Request* (sur Github ou Bitbucket)

3.52.4 Alice accepte directement le Merge Request

Alice peut accepter automatiquement le Merge Request de Bob si celui-ci est très simple, directement sur le Gitlab. Dans ce cas elle fait ensuite un simple *pull* (git pull) de son master pour que sa copie locale soit à jour aussi.

Sinon Alice va tester le code de Bob. Pour celà :

3.52.5 Alice récupère la branche de Bob

```
git fetch bob bd
```

ou

```
git fetch bob bd --tags
```

pour aussi récupérer des tags éventuels.

3.52.6 Alice consulte la branche de Bob

```
git checkout Bob/bd
```

(pour lister toutes les branches : `git branch -av`)

3.52.7 Alice revient dans son master

```
git checkout master
```

3.52.8 puis Alice accepte le Merge Request et met à jour sa copie locale

```
git pull
```

3.53 Bob

Il reste à Bob à récupérer le master d'Alice pour se mettre à jour :

3.53.1 Soit avec un fetch et un merge

```
git fetch alice master
git merge alice/master
```

3.53.2 Soit directement avec un pull

```
git pull alice master
```

3.54 Commandes utiles avec les branches

3.54.1 Renommage de branche

```
git branch -m <oldname> <newname>
```

3.54.2 Modification d'une branche distante

Si vous voulez mettre à jour une branche distante *mabranche* :

```
git push origin mabranche --force
```

A utiliser si d'autres n'ont pas déjà intégré votre branche ...

3.54.3 Suppression d'une branche distante

```
git push origin --delete mabranche
```

3.55 Merge vs Rebase

3.55.1 Illustration

Soit le petit scénario suivant avec 5 commits pour voir concrètement la différence entre Merge et Rebase

3.55.2 Merge

```
echo 1 > Readme.md
git add Readme.md
git commit -m "un"
echo 2 > Readme.md
git commit -am "deux"

git checkout -b feature
echo 3 > feature.md
git add feature.md
git commit -m "trois"

git checkout master
echo 4 > Readme.md
git commit -am "quatre"

git merge feature --no-edit

echo 5 > Readme.md
git commit -am "cinq"

git log --graph --pretty='format:%h %an (%ar) %s' --abbrev-commit
```

```
* 8161013 Gerard (il y a 2 jours) cinq
*   a6deee3 Gerard (il y a 2 jours) Merge branch 'feature'
|\
| * 6c0bcf0 Gerard (il y a 2 jours) trois
* | 881c431 Gerard (il y a 2 jours) quatre
|/
* 88d5081 Gerard (il y a 2 jours) deux
* d761e62 Gerard (il y a 2 jours) un
```

3.55.3 Rebase

```
echo 1 > Readme.md
git add Readme.md
git commit -m "un"
echo 2 > Readme.md
git commit -am "deux"

git checkout -b feature
echo 3 > feature.md
git add feature.md
git commit -m "trois"

git checkout master
echo 4 > Readme.md
```

(suite sur la page suivante)

(suite de la page précédente)

```
git commit -am "quatre"

git rebase feature
git log --graph --pretty=oneline --abbrev-commit

echo 5 > Readme.md
git commit -am "cinq"

git log --graph --pretty='format:%h %an (%ar) %s' --abbrev-commit
```

```
* 824e964 cinq
* 5164ca2 quatre
* 4905a5f trois
* f0b3d04 deux
* 2d84cee un
```

merge vs rebase (https ://www.atlassian.com/git/tutorials/merging-vs-rebasing) en français (https ://www.atlassian.com/fr/git/tutorials/merging-vs-rebasing)

3.56 Utilisation de Merge

L'utilisation la plus courante est :

```
git checkout master
git merge feature
```

Si tout se passe sans encombre, parfait ! Sinon on est dans les ennuis :

```
Auto-merging Document
CONFLICT (content): Merge conflict in mon-prog.py
Automatic merge failed; fix conflicts and then commit the result.
```

On peut alors editer le fichier et régler le conflit *à la main* ou avec un outil comme kdiff3 ou vscode.

Sinon, si on sait que la version correcte est dans master :

```
git checkout --ours mon-prog.py
```

Si au contraire, on veut garder la version de la branche feature :

```
git checkout --theirs mon-prog.py
```

puis dans tous les cas :

```
git add mon-prog.py
git merge --continue
```

Le conflit est résolu !

3.57 Utilisation de Rebase

Le merge est souvent le plus utilisé (Merge request, etc.) mais il y a quelques utilisations importantes du rebase dont les 2 suivantes :

3.57.1 pour se mettre à jour sur un projet sur lequel on collabore

Si Bob a réalisé une feature dans une branche *new_feature* dans un projet auquel il collabore et que le projet évolue sensiblement pendant la réalisation de cette feature, Bob va devoir se remettre à jour du projet pour cela il fera usage du rebase comme ceci par exemple :

```
git checkout master
git pull depot-reference master
git checkout new_feature
git rebase master
```

3.57.2 pour revisiter un historique défaillant

Les historiques Git sont rarement parfaits du premier coup alors il y a parfois nécessité de *revisiter le passé* et de faire un nettoyage. La machine à remonter le temps en Git s'appelle le *rebase interactif* On choisit le commit à partir duquel on veut opérer des changements. On note l'id du commit précédent et on demande à notre machine à remonter le temps de nous conduire à ce commit :

```
git rebase -i d5ea32
```

Un écran s'ouvre alors vous permettant d'éditer, supprimer, déplacer ou rejouer tel quel un commit. Si on demande à éditer le commit, il faudra généralement le défaire `git reset HEAD^`, procéder aux modifications puis relancer le processus avec un `git rebase --continue`

Voir la doc (https ://git-scm.com/book/fr/v2/Les-branches-avec-Git-Rebaser-Rebasing) pour plus de détails sur rebase.

3.58 Configuration PHP

3.58.1 Le fichier PHP.ini

Le fichier PHP.ini contient toutes les directives essentielles de réglage.

- Taille des fichiers téléchargeables
- Safe-Mode
- Affichage et traitement des erreurs

— Communication avec MySQL

Danger : Attention, les directives de ce fichier sont très importantes pour la sécurité d'un serveur en production. Il faudra y veiller et les vérifier minutieusement dans ce cas. Sous certaines distributions de Linux, il existe 2 versions de ce fichier une de developpement et une autre pour un serveur de production. N'oubliez pas d'activer la bonne version selon le contexte et de la vérifier en tous les cas.

3.58.2 Sous la plupart des distributions Linux :

2 versions du PHP.ini sont fournies :
— Une version de dev
— Une version de prod
Il faut choisir entre les 2 en faisant le bon lien symbolique

3.58.3 Les directives principales PHP.ini :

Ces directives sont très nombreuses. J'ai retenu les plus importantes dans le fichier suivant en commentant leur rôle.

```
[PHP]

;;;;;;;;;;;;;;;;;;;
; About php.ini   ;
;;;;;;;;;;;;;;;;;;;
; Fichier de configuration principal de PHP
; qui permet de préciser les principales options
; Sous certaines distributions Linux, il en existe 2 versions:
; une de developpement et une autre pour un serveur de production

;;;;;;;;;;;;;;;;;;;;
; Language Options ;
;;;;;;;;;;;;;;;;;;;;

; Pour activer PHP
engine = On

; On peut mettre à faux : les tags <? .... ?> ne sont pas reconnus.
short_open_tag = Off

; Allow ASP-style <% %> tags.
; http://php.net/asp-tags
asp_tags = Off

; The number of significant digits displayed in floating point␣
↪numbers.
```

(suite sur la page suivante)

(suite de la page précédente)

```
; http://php.net/precision
precision = 14

; Compatibilité an 2000
y2k_compliance = On

; Taille des buffers
output_buffering = 4096

; Pour servir ou pas des pages compressées
zlib.output_compression = Off

; Mettre à On pour forcer les flush en phase de debuggage
implicit_flush = Off


; Safe Mode
; http://php.net/safe-mode
; On peut le laisser désactivé car
; a été déclaré OBSOLETE depuis PHP 5.3
Safe_mode = Off


;Pour désactiver certaines fonctions PHP
; indépendant du safe_mode
; http://php.net/disable-functions
disable_functions =

; meme chose avec des classes
disable_classes =

; Colors for Syntax Highlighting mode.
; A utiliser avec la fonction highlight_file() = show_source()
Highlight.string  = #DD0000
highlight.comment = #FF9900
highlight.keyword = #007700
highlight.bg      = #FFFFFF
highlight.default = #0000BB
highlight.html    = #000000

;;;;;;;;;;;;;;;;;
; Miscellaneous ;
;;;;;;;;;;;;;;;;;

; On peut l'enlever sur un serveur de production
; mais n'est pas une menace de sécurité
expose_php = On

;;;;;;;;;;;;;;;;;;;
```

(suite sur la page suivante)

(suite de la page précédente)

```
; Resource Limits ;
;;;;;;;;;;;;;;;;;;;

;Temps d'éxécution max d'un script
;Attention si vous avez du code un peu long à s'éxécuter !
max_execution_time = 30

; Traitement des données reçues
; laisser la valeur par défaut
max_input_time = 60

; Taille mémoire maxi donnée à un script PHP
memory_limit = 128M

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Error handling and logging ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; Pour un serveur de Production: E_ALL & ~E_DEPRECATED
; Pour un serveur de développement
error_reporting = E_ALL | E_STRICT

; Affichage des erreurs sur la sortie standard
; cad sur le navigateur
; A désactiver sur un serveur de production
; Utile pour un développeur
display_errors = On

; Affichage des erreurs au démarrage de PHP
; Pour débugger des erreurs sur des plugins
; ou des modules complémentaires de PHP
display_startup_errors = Off

; logger les erreurs
; A activer
log_errors = On

; Set maximum length of log_errors. In error_log information about␣
 ↪the source is
; added. The default is 1024 and 0 allows to not apply any maximum␣
 ↪length at all.
; http://php.net/log-errors-max-len
log_errors_max_len = 1024

;Ne pas répéter les erreurs identiques
ignore_repeated_errors = On

; ... sauf si elles proviennent de fichiers différents
ignore_repeated_source = Off
```

(suite sur la page suivante)

(suite de la page précédente)

```
; Rapporter les fuites de mémoire
; A activer en phase de dévelopement
report_memleaks = On

; La variable $php_errormsg
; contiendra le texte du dernier message
; d'erreur
; A désactiver sur un serveur de production
track_errors = On

; http://php.net/html-errors
html_errors = On

; A faire pointer sur une copie locale de la documentation
; de PHP
; A désactiver sur un serveur de production
docref_root = "/docs/php/"

; Extension des fichiers de documentation
docref_ext = .html

; Chaine à afficher avant un message d'erreur
; Ici pour qu'il s'affiche en rouge
; Réservé aux serveurs de dévelopement
error_prepend_string = "<font color=#ff0000>"

; Fermeture du tag précédent
error_append_string = "</font>"

; Pour changer le fichier où sont loggées
; les erreurs. Laisser inchangé sauf
; cas particulier
;error_log = syslog

;;;;;;;;;;;;;;;;;
; Data Handling ;
;;;;;;;;;;;;;;;;;

; The separator used in PHP generated URLs to separate arguments.
; PHP's default setting is "&".
; http://php.net/arg-separator.output
; Example:
;arg_separator.output = "&amp;"

; List of separator(s) used by PHP to parse input URLs into␣
↪variables.
; PHP's default setting is "&".
; NOTE: Every character in this directive is considered as␣
↪separator!
```

(suite sur la page suivante)

(suite de la page précédente)

```
; http://php.net/arg-separator.input
; Example:
;arg_separator.input = ";&"

; This directive determines which super global arrays are␣
↪registered when PHP
; starts up. If the register_globals directive is enabled, it also␣
↪determines
; what order variables are populated into the global space. G,P,C,E␣
↪& S are
; abbreviations for the following respective super globals: GET,␣
↪POST, COOKIE,
; ENV and SERVER. There is a performance penalty paid for the␣
↪registration of
; these arrays and because ENV is not as commonly used as the␣
↪others, ENV is
; is not recommended on productions servers. You can still get␣
↪access to
; the environment variables through getenv() should you need to.
; Default Value: "EGPCS"
; Development Value: "GPCS"
; Production Value: "GPCS";
; http://php.net/variables-order
variables_order = "GPCS"

; laisser la valeur par défaut
request_order = "GP"

; Ca fait longtemps qu'il faut garder cette directive à Off
register_globals = Off

; Determines whether the deprecated long $HTTP_*_VARS type␣
↪predefined variables
; are registered by PHP or not. As they are deprecated, we␣
↪obviously don't
; recommend you use them. They are on by default for compatibility␣
↪reasons but
; they are not recommended on production servers.
; Default Value: On
; Development Value: Off
; Production Value: Off
; http://php.net/register-long-arrays
register_long_arrays = Off

; A activer seulement si vous voulez utiliser PHP
; en ligne de commande et lui passer des arguments
register_argc_argv = Off

; Meilleure performance avec :
```

(suite sur la page suivante)

(suite de la page précédente)

```
auto_globals_jit = On

; Taille maximale des données acceptées en POST
; http://php.net/post-max-size
post_max_size = 8M

; A éviter désormais
magic_quotes_gpc = Off

; idem
magic_quotes_runtime = Off

; mimetype par défaut : text/html
default_mimetype = "text/html"

; Jeu de caractères par  défaut
; laisser à vide ou choisir un jeu de caractères
;default_charset = "iso-8859-1"
default_charset = "utf-8"

;;;;;;;;;;;;;;;;
; File Uploads ;
;;;;;;;;;;;;;;;;

;
; Autoriser les "uploads" de fichiers
file_uploads = On

; Spécifier le répertoire temporaire por les fichiers
; uploadés :
;upload_tmp_dir = /tmp/upload-dir

; Taille maxi pour les fichies uploadés
upload_max_filesize = 4M

; Nbre de fichiers maxi pouvant être uploadés en une seule requête
max_file_uploads = 20

;;;;;;;;;;;;;;;;;;
; Fopen wrappers ;
;;;;;;;;;;;;;;;;;;

; Whether to allow the treatment of URLs (like http:// or ftp://)␣
 ↪as files.
; http://php.net/allow-url-fopen
allow_url_fopen = On

; Whether to allow include/require to open URLs (like http:// or␣
 ↪ftp://) as files.
```

(suite sur la page suivante)

(suite de la page précédente)

```
; http://php.net/allow-url-include
allow_url_include = Off

; Define the anonymous ftp password (your email address). PHP's␣
↪default setting
; for this is empty.
; http://php.net/from
;from="john@doe.com"

; Define the User-Agent string. PHP's default setting for this is␣
↪empty.
; http://php.net/user-agent
;user_agent="PHP"

; Timeout pour les flux basés sur des sockets
default_socket_timeout = 60

;;;;;;;;;;;;;;;;;;;;;
; Dynamic Extensions ;
;;;;;;;;;;;;;;;;;;;;;

; ; Sous Windows:
;   extension=msql.dll
; ... et sous UNIX:
;
;   extension=msql.so
;
; ... ou avec un chemin:
;
;   extension=/path/to/extension/msql.so
;

;;;;;;;;;;;;;;;;;;;
; Module Settings ;
;;;;;;;;;;;;;;;;;;;

[Date]
; Fuseau horaire utilisé
date.timezone ="Europe/Paris"

[iconv]
; conversion d'un système d'encodage à un autre
;iconv.input_encoding = ISO-8859-1
;iconv.internal_encoding = ISO-8859-1
;iconv.output_encoding = ISO-8859-1

[Pdo_mysql]
; En cas d'utilisation du nouveau moteur  mysqlnd
pdo_mysql.cache_size = 2000
```

(suite sur la page suivante)

(suite de la page précédente)

```
; Socket par défaut pour la connexion à MySQL
; La valeur par défaut fonctionne le plus souvent
pdo_mysql.default_socket=/var/mysql/mysql.sock

[mail function]
; For Win32 only.
; http://php.net/smtp
SMTP = localhost
; http://php.net/smtp-port
smtp_port = 25

; Emplacement pour logger les appels à la fonction mail()
;mail.log =

[MySQL]
; Autorise les connexions persistantes
; N'apporte AUCUNE fonctionnalité supplémentaire
; Mais peut améliorer les performances
mysql.allow_persistent = On

; If mysqlnd is used: Number of cache slots for the internal result␣
↪set cache
; http://php.net/mysql.cache_size
mysql.cache_size = 2000

; Nbre maxi de liens persistants
mysql.max_persistent = -1

; Nombre maxi de liens permanents :
; -1 veut dire sans limitation
Mysql.max_links = -1

; Port par défaut de MySQL
mysql.default_port = 3306

; Laisser généralement la valeur par défaut
mysql.default_socket =

; Hôte par défaut pour mysql_connect()
mysql.default_host =

; Utilisateur  par défaut pour mysql_connect()
mysql.default_user =

; Passwd  par défaut pour mysql_connect()
; Ce n'est pas une bonne chose de garder
; le passwd ici !! obsolete
mysql.default_password =
```

(suite sur la page suivante)

(suite de la page précédente)

```
; Timeout de connexion à MySQL
mysql.connect_timeout = 60

; Mode de débuggage MySQL
mysql.trace_mode = Off

[MySQLi]
; Nbre maxi de liens persistants
mysqli.max_persistent = -1

; Autorise les connexions persistantes
; N'apporte AUCUNE fonctionnalité supplémentaire
; Mais peut améliorer les performances
mysqli.allow_persistent = On

; Maximum number of links.  -1 means no limit.
; http://php.net/mysqli.max-links
mysqli.max_links = -1

; If mysqlnd is used: Number of cache slots for the internal result␣
↪set cache
; http://php.net/mysqli.cache_size
mysqli.cache_size = 2000

; Por pour mysqli
mysqli.default_port = 3306

; Socket par défaut pour MySQLi
mysqli.default_socket = /tmp/mysql.sock

; Autorise ou interdit la reconnexion
mysqli.reconnect = Off

[mysqlnd]
; activation des statistiques de mysqlnd
; a des fins de réglages du serveur de BD
mysqlnd.collect_statistics = On

; Même chose avec les opérations sur la mémoire
mysqlnd.collect_memory_statistics = Off

; Size of a pre-allocated buffer used when sending commands to␣
↪MySQL in bytes.
; http://php.net/mysqlnd.net_cmd_buffer_size
;mysqlnd.net_cmd_buffer_size = 2048

; Size of a pre-allocated buffer used for reading data sent by the␣
↪server in
```

(suite sur la page suivante)

(suite de la page précédente)

```
; bytes.
; http://php.net/mysqlnd.net_read_buffer_size
;mysqlnd.net_read_buffer_size = 32768

[bcmath]
; Number of decimal digits for all bcmath functions.
; http://php.net/bcmath.scale
bcmath.scale = 0

[Session]
; .../...

; Les sessions doivent-elles utiliser les cookies ?
session.use_cookies = 1

; Envoyer les cookies à travers
; des connexions sécurisées
; le défaut est Off
;Session.cookie_secure =

; PHP maintient un cookie avec l'identifiant de session
; c'est une précaution visant à éviter
; le vol de session
; ce n'est pas une parade absolue
session.use_only_cookies = 1

; Nom de la session
session.name = PHPSESSID

; Démarrage automatique de session
; Désactivé par défaut
session.auto_start = 0

; Durée de vie du cookie
; Si placé à 0, le temps que le navigateur
; soit redémarré
session.cookie_lifetime = 0

; Domaine de validité du cookie
session.cookie_domain =

; Pour interdire à javascript d'accéder à ce cookie
session.cookie_httponly = On

;
; HTTP_REFERER doit contenir cette sous-chaine
; pour être considéré comme valide
Session.referer_check =
```

(suite sur la page suivante)

(suite de la page précédente)

```
; Durée d'expiration du document en minutes
session.cache_expire = 180

; Choix d'une fonction de hachage pour les sessions
; comme :
;   0  (MD5 128 bits)
;   1  (SHA-1 160 bits)
session.hash_function = 0

[Assertion]
; Assertions actives (défaut)
assert.active = On

; Emettre un warning en cas d'assertion non vérifiée
assert.warning = On

; S'arrêter en cas d'assertion non satisfaite
; Désactivé par défaut
;assert.bail = Off

; Fonction utilisateur à appeller en cas d'assertion non satisfaite
;assert.callback = 0
```

CHAPITRE 4

GIT

Tout bon développeur doit aujourd'hui savoir utiliser un système de gestion de version pour son code et pour collaborer. Git est aujourd'hui le plus répandu. Vous trouverez à la fin de ce cours un rappel des principales commandes git pour démarrer :

GIT start

et quelques commandes pour travailler à plusieurs sur un projet avec les branches git et des *Merge Request* (ou Pull Requests) :

GIT branches

Enfin, quelques compléments sur Merge et Rebase :

GIT branches

Un cours complet sur Git et des fiches mémo en pdf (https ://www.univ-orleans.fr/iut-orleans/informatique/intra/tuto/git/slides.html)

CHAPITRE 5

Configuration

Quelques compléments sur le fichier de configuration PHP.ini :

GIT branches

CHAPITRE 6

Références

— Manuel PHP (http ://php.net/manual/fr/)
— intro JS (https ://www.univ-orleans.fr/iut-orleans/informatique/intra/tuto/js-initiation/slides1.html#/le-langage-javascript)
— TDD (en anglais) (https ://roza.gitlab.io/git-tdd-en)

CHAPITRE 7

Index et recherche

— genindex
— search

Index