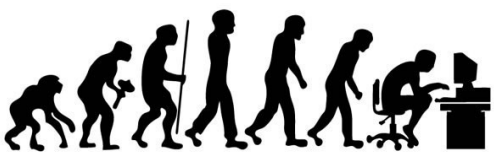
ICT -133b

Réaliser des applications Web en Session-Handling



Auteur :	Laurent Moine
Date :	12.03.18
Nom du document :	ict133b_v2.0.odt
N° de révision:	6

Table des matières

1 Objectifs opérationnels du module ICT-133b

Selon le plan modulaire de ICT Switzerland[1] , les objectifs à atteindre pour le module ict133 dans sa version 3.0 sont les suivants:

1) Analyser la donnée, projeter la fonctionnalité et déterminer le concept de la réalisation.

1.1 Connaître des solutions possibles afin de délimiter les fonctionnalités côté client comme côté serveur en se basant sur les directives.

1.2 Connaître les éléments d'un concept de réalisation pour des applications Web.

1.3 Connaître les avantages de la séparation entre la présentation et la logique du programme ainsi que leur mise en œuvre dans une application Web. (chap 9)

2) Réaliser une fonctionnalité spécifique d'une application Web par Session-Handling, authentification et vérification de formulaire.

1.1 Connaître des éléments de formulaires ainsi que des fonctions pour vérifier les saisies de l´utilisateur, et comment ceux-ci mis en œuvre conformément aux directives. (chap 10)

1.2 Connaître des techniques courantes pour la réalisation du suivi de session. (chap 13)

1.3 Connaître les risques sur la sécurité et des solutions possibles pour protéger une application Web. (chap 10.3)

1.4 Connaître des possibilités d'administration de lots de données.

3) Programmer une application Web à l'aide d'un langage de programmation compte tenu des exigences liées à la sécurité.

1.2 Connaître des langages de programmation possibles pour des applications Web.

1.3 Connaître des solutions possibles pour la mise en œuvre d'architectures appropriées pour des applications Web. (chap 2.1)

4) Vérifier la fonctionnalité et la sécurité de l'application Web à l'aide du plan tests, verbaliser les résultats et, le cas échéant, corriger les erreurs..

1.1 Connaît des procédures de tests, et leur contribution, pour garantir la qualité d'applications Web.

1.2 Connaître des procédures de tests permettant de vérifier les solutions possibles contre les Cross-Site-Scripting, Script-Injection et Session-Hijacking. (chap 11)

1 https://cf.ict-berufsbildung.ch/modules.php?name=Mbk&a=20101&cmodnr=133&noheader=1

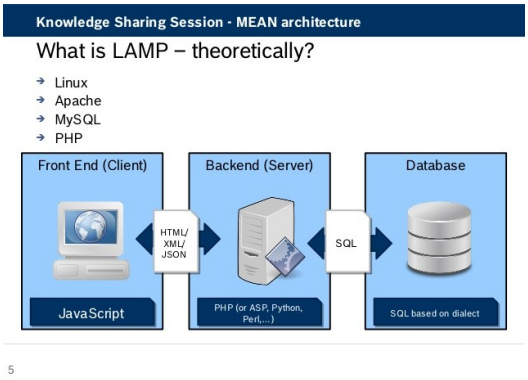
2 Introduction

Dans le contexte du web, la logique applicative située sur le serveur prend en charge l'accès aux bases de données. Elle doit pouvoir ouvrir une session sur un serveur de bases de données. Elle doit aussi être en mesure d'exécuter des requêtes de manipulation des données.

Si des scripts écrits en PHP pilotent la logique applicative, le serveur de bases de données associées sera probablement MySQL, bien que ce ne soit pas une obligation.

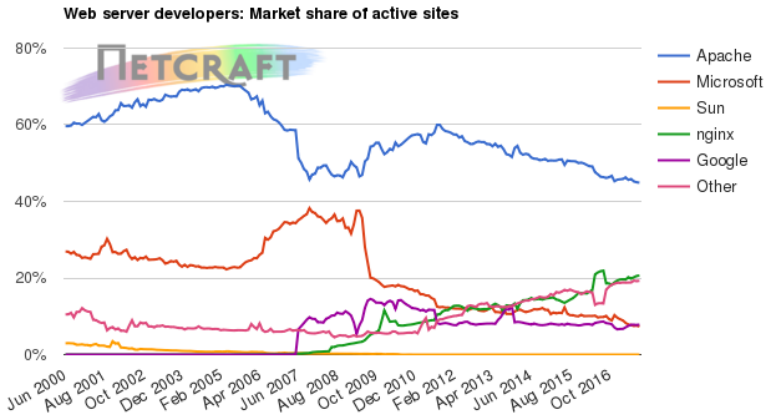
2.1 Architecture du système

L'architecture usuelle dans un cas général est la suivante



Le front-end qui s'exécute sur le client (souvent un navigateur). Il est écrit en HTML et peut-être en Javascript. il communique via le protocole HTTP, avec les scripts (souvent php, asp, python, ...) implémentés sur le serveur web[2] (souvent Apache, IIS ou ngnix). Ces mêmes scripts accèdent au serveur de base de données (MySql, Oracle, ...) qui est souvent localisé sur le même serveur.

Le but de ce cours est de décortiquer les interactions entre les scripts côté serveur et le Front End.



2 https://news.netcraft.com/archives/2017/09/11/september-2017-web-server-survey.html

3 Fichiers scripts

Un script PHP peut contenir du code PHP et du code HTML dans des proportions diverses

3.1 pré-requis

" Nous présumons que vous avez un serveur web avec le support PHP activé, et que les fichiers terminés par l'extension .php sont traités par PHP. (...)

Si votre serveur web supporte PHP, vous n'avez rien à faire. Simplement, créez un dossier, puis créez un fichier texte, avec l'extension .php : le serveur va automatiquement l'exécuter avec PHP. Il n'y a pas de compilation, ou d'installation compliquée. Gardez en tête que les fichiers sont comparables à des fichiers HTML, dans lesquels vous allez utiliser des balises magiques, qui feront beaucoup de choses pour vous."

Extrait du manuel PHP

Pour que ça marche :

Le fichier DOIT avoir .php comme extension

Le fichier DOIT se trouver dans le répertoire géré par le serveur

3.2 Script pur

L'exemple ci-dessous ne contient que du php. Le code HTML est généré par l'interpréteur php.

```php
<?php
/**
 * Created by PhpStorm.
 * User: lmo
 * Date: 05.12.16
 * Time: 15:41
 */
echo '<!DOCTYPE html ';
echo '<html lang="fr">';
echo '<head>';
echo '  <meta charset="utf-8" />';
echo '  <meta name="DCTERMS.creator" content="Prénom Nom" />';
echo '  <meta name="DCTERMS.subject" content="Thème" />';
echo '  <meta name="DCTERMS.created" content="aaaa-mm-jj" />';
echo '  <title>Exercices X</title>';
echo '</head>';
echo '<body>';
echo '  <p>Exemple de page entièrement générée par code php</p>';
echo '</body>';
echo '</html>';
?>
```

La fonction echo envoie dans le flux standard de sortie, ici le navigateur web, la chaîne de caractères comprise entre les apostrophes. C'est dans cette chaîne que le programmeur fait

apparaître les balises html nécessaires.

3.3 Script mélangé

L'exemple ci-dessous montre qu'il est possible d'écrire des fichiers de scripts php contenant des balises html au milieu desquelles se retrouvent des parties de code php.

```
<?php
/**
 * Created by PhpStorm.
 * User: lmo
 * Date: 05.12.16
 * Time: 15:58
 */
?>
<!DOCTYPE html >
 <html lang="fr">
 <head>
   <meta charset="utf-8" />
   <meta name="DCTERMS.creator" content="Prénom Nom" />
   <meta name="DCTERMS.subject" content="Thème" />
   <meta name="DCTERMS.created" content="aaaa-mm-jj" />
   <title>Exercices X</title>
 </head>
<body>
  <?php
    echo '<p>Ce texte est généré par code php</p>';
  ?>
  <h3>Des liens en php</h3>
  <a href="<?='http://www.php.net/manual/fr' ?>" target="_blank">
      manuel du PHP en ligne
  </a>
</body>
</html>
```

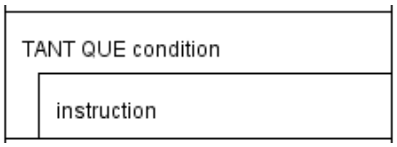
Dans ce fichier script, toute la partie html est retournée telle quelle au navigateur. Seule la partie comprise entre les balises <?php ?> sera interprétée, c'est-à-dire, traduite en code html.

La balise de définition du lien HTML montre que l'instruction echo peut simplement être remplacer par le signe = directement après l'ouverture de la balise PHP

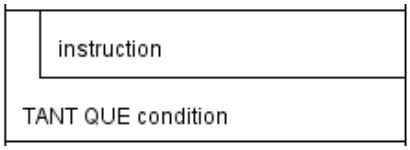
4 Base de la syntaxe du PHP

La syntaxe du langage est tirée directement de la syntaxe du langage C. Les programmeurs connaissant ce langage seront donc rapidement efficaces.

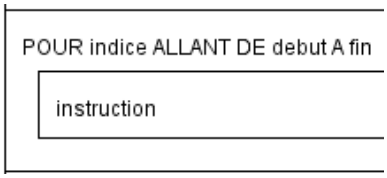
4.1 Les boucles




```
while( condition ){
  instruction;
}
```

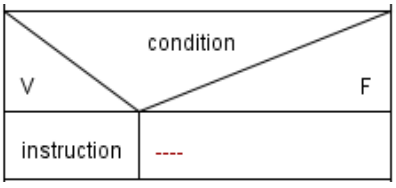



```
do{
  instruction;
} while( condition )
```

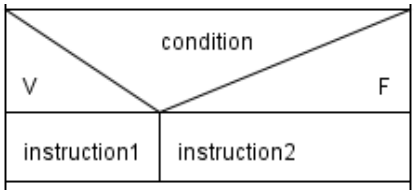



```
for($i=$debut;
     $i<=fin;i++){
  instruction;
}
```

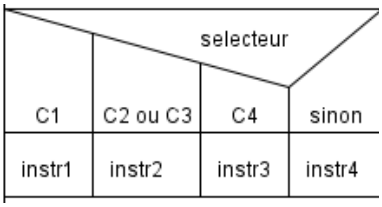
4.2 Les tests




```
if( condition )
{
  instruction;
}
```




```
if( condition )
{
  instruction1;
}
else
{
  instruction1;
}
```




```
switch(selecteur)
{
  case C1 : instr1;
            break;
  case C2 :
  case C3 : instr2;
            break;
  case C4 : instr3;
            break;
  default : instr4;
            break;
}
```

4.3 Les commentaires

Commentaires de fin de ligne

```
instruction; // Ceci est un commentaire
```

Commentaire multi ligne (bloc)

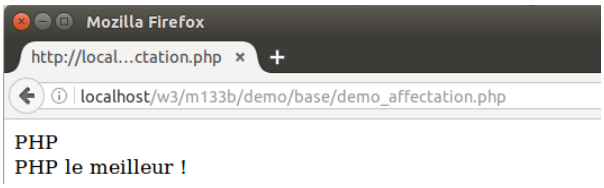
```
/*
  Commentaire ligne 1
  Commentaire ligne 2
/*
```

4.4 L'affichage

L'envoi d'une réponse au navigateur ou l'affichage à la console se font à l'aide de l'instruction **echo**[3].

Cette instruction permet l'affichage de plusieurs valeurs. Il suffit de les séparer par des virgules.

```
echo 'PHP';
echo '</br>';
// La virgule ',' permet de séparer les fragments de chaînes de caractères envoyés à
echo.
echo 'PHP', ' le meilleur !';
```



4.5 Les variables

On déclare une variable en lui attribuant un identifiant valide[4] (commençant soit par un caractère alphabétique, soit par le caractère souligné '_') préfixé du signe dollar '$', et en lui assignant une valeur.

Sauf si elle est déclarée au sein d'une définition de fonction, la portée d'une variable est globale (elle concernera le script entier à partir de la déclaration, ou de la redéclaration de la variable).

```
$titre  = '';
```

4.6 Les constantes

Une constante[5] est un identifiant (un nom) qui représente une valeur simple. Comme son nom le suggère, cette valeur ne peut jamais être modifiée durant l'exécution du script.

Vous pouvez définir une constante en utilisant la fonction define() ou en utilisant le mot-clé *const* [6]en dehors d'une définition de classe à partir de PHP 5.3.0. Une fois qu'une constante est définie, elle ne peut jamais être modifiée, ou détruite.

```
const   USER_NAME =  'Paul Carbone';
define('USER_TITLE', 'Formateur');
```

3 http://php.net/manual/fr/function.echo.php
4 http://php.net/manual/fr/language.variables.basics.php
5 http://php.net/manual/fr/language.constants.php
6 http://php.net/manual/fr/language.constants.syntax.php

4.7 Types de données

Il n'est pas nécessaire de déclarer le type des données que l'on manipule, celui-ci est affecté dynamiquement par PHP.

En revanche, il n'est pas inutile de connaître les différents types possibles. PHP en propose huit différents[7].

Booléens[8]

Un booléen (cf. les expressions) exprime une valeur VRAIE (true), ou FAUSSE (false).

Valeur booléenne	Constante PHP	Conversion des numérique
VRAI	TRUE ou true	Différent de 0 (≠ 0)
FAUX	FALSE ou false	Égal à 0 (= 0)

Entiers[9]

Un entier (integer, ou int) est un nombre de l'ensemble des entiers naturels {..., -2, -1, 0, 1, 2, ...}.

On peut l'exprimer de trois manières :

```
$a = 1234;  // Nombre entier en base décimale soit 1234.

$a = 01234; // En base octale, préfixé 0 (zéro) soit 668 en décimale.
$a = 0x1234;// En base hexa, préfixé 0x soit 4660 en décimale.
```

Réels

Un nombre réel (float, ou double), appelé aussi *à virgule flottante*, peut être exprimé de deux manières :

```
$a = 1.234;
$a = 1.234e3; // Notation scientifique, 1.234*10^3 = 1234 en décimal.
```

Chaînes de caractères[10]

Une chaîne de caractères doit être encadrée par des apostrophes ou des guillemets :

- apostrophes (' ') si elle ne contient rien qui nécessite d'être échappé (comme un \n par exemple);

```
$variable = 'PHP le meilleur !';
echo '1) Et l\'ami dit " = $variable \n"<br> ';
```

- guillemets (" ") si l'on veut que les séquences d'échappement et les variables qu'elle contient soient interprétées (on parle de *substitution de variables*) :

7 http://php.net/manual/fr/language.types.intro.php
8 http://php.net/manual/fr/language.types.boolean.php
9 http://php.net/manual/fr/language.types.integer.php
10 http://php.net/manual/fr/language.types.string.php

```php
$variable = 'PHP le meilleur !';
echo "2) Et l'ami dit \" = $variable \" \n<br> ";
// Autre possibilité.
echo '3) Contenu de la variable = ', $variable, '<br>';
```

L'interprétation des lignes ci-dessus par le navigateur montre bien dans **l'exemple 1**, que la chaîne de caractère définie avec apostrophes ne réalise pas la substitution des variables

Le code source de la page montre aussi que la chaîne de caractère définie avec les apostrophes n'interprète pas les échappement. Le retour à la ligne de **l'exemple 1** n'est pas interprétée.

```
1) Et l'ami dit " = $variable \n"<br> 2) Et l'ami dit " = PHP le meilleur ! "
<br> 3) Contenu de la variable = PHP le meilleur !<br>
```

4.8 L'affectation

On affecte une valeur à une variable[11] à l'aide du signe égal = :

```php
// Affecte la valeur entière 3 à $a.
$a = 3;
```

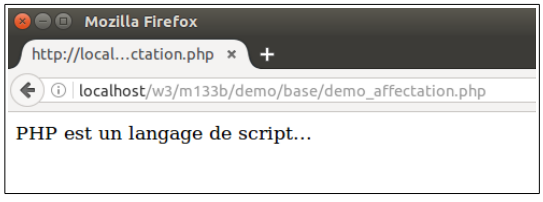
Une variable peut être réaffectée; sa valeur précédente sera chaque fois écrasée par la nouvelle valeur qu'on lui aura assignée :

```php
$a = 123;
$a = 'PHP';
// La virgule ',' permet de séparer les fragments de chaînes de caractères
// envoyés à echo.
echo $a,' le meilleur !';
```

Assignation par référence

Le signe **&** permet d'assigner la valeur d'une variable source à une variable destination. La variable de destination devient un synonyme de la variable source

```php
$a = 'PHP';
$refA = &$a;
echo $refA, ' est un langage de script...';
```



11 http://php.net/manual/fr/language.operators.assignment.php

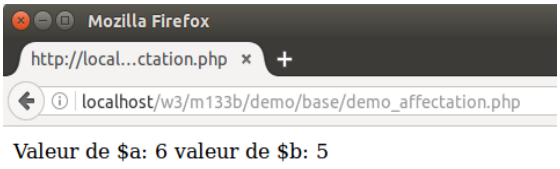
Concaténation[12]

Le point **.** permet de concaténer variables et chaînes de caractères ou plusieurs chaînes :

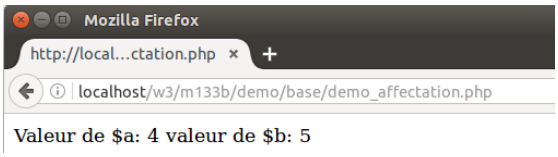
```
$a = 'PHP est un langage de script ';
$b = 'JS aussi...';
echo $a.$b;
```

Post-incrémentation/décrémentation[13]

```
$a = 5;
// Assigne la valeur de $a à $b, puis incrémente $a.
$b = $a++;
```

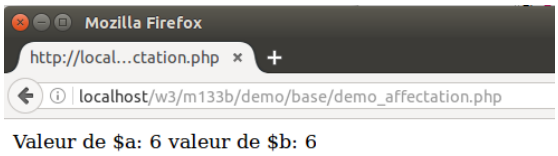



```
$a = 5;
// Assigne la valeur de $a à $b, puis décrémente $a.
$b = $a--;
echo 'Valeur de $a: ', $a, ' valeur de $b: ', $b, '</br>';
```



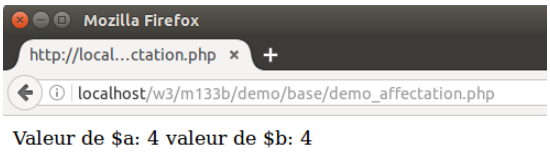
Pré-incrémentation/décrémentation

```
$a = 5;
// Incrémente $a puis assigne la valeur de $a à $b.
$b = ++$a;
echo 'Valeur de $a: ', $a, ' valeur de $b: ', $b, '</br>';
```




```
$a = 5;
// Décrémente $a puis sssigne la valeur de $a à $b.
$b = --$a;
echo 'Valeur de $a: ', $a, ' valeur de $b: ', $b, '</br>';
```

12 http://php.net/manual/fr/language.operators.string.php
13 http://php.net/manual/fr/language.operators.increment.php



On peut utiliser des raccourcis pour affecter une valeur et le résultat d'une opération arithmétique à une variable en même temps :

```php
// Correspond à l'instruction $a = $a + 5.
$a += 5;

// Correspond à l'instruction $a = $a - 5.
$a -= 5;

// Correspond à l'instruction $a = $a * 5.
$a *= 5;

// Correspond à l'instruction $a = $a / 5.
$a /= 5;

// Correspond à l'instruction de concaténation $a = $a .'j'aime' .
$a = 'PHP';
$a .= ' j\'aime';
```

4.9 Les opérateurs

Opérateurs arithmétiques[14]

signe	exemple	description
+ (addition)	`$res = $a + $b;`	additionne $a et $b
- (soustraction)	`$res = $a + $b;`	soustrait $b de $a
* (multiplication)	`$res = $a * $b;`	multiplie $a par $b
/ (division)	`$res = $a / $b;`	divise $a par $b
% (modulo)	`$res = $a % $b;`	reste de la division de $a par $b

Opérateurs de comparaison[15]

signe	exemple	retourne TRUE si :
== (égal)	`$res = $a == $b`	$a est égal à $b ('12'==12 → true)
=== (identique)	`$res = $a === $b;`	$a est égal et du même type que $b ('12'===12 → false)
!= (différent)	`$res = $a != $b;`	$a est différent de $b

14 http://php.net/manual/fr/language.operators.arithmetic.php
15 http://php.net/manual/fr/language.operators.comparison.php

!== (pas identique)	`$res = $a !== $b;`	$a est différent de $b ou s'ils ne sont pas du même type
< (inférieur)	`$res = $a < $b;`	$a est inférieur à $b
> (supérieur)	`$res = $a > $b;`	$a est supérieur à $b
<= (inférieur ou égal)	`$res = $a <= $b;`	$a est inférieur ou égal à $b
>= (supérieur ou égal)	`$res = $a >= $b;`	$a est supérieur ou égal à $b

Opérateurs logiques[16]

signe	exemple	retourne TRUE si :
&& (et)	`res = $a && $b;`	$a et $b retournent TRUE
AND (et)	`$res = $a and $b;`	$a et $b retournent TRUE *
|| (ou)	`$res = $a || $b;`	$a ou $b retourne TRUE
OR (ou)	`$res = $a or $b;`	$a ou $b retourne TRUE *
XOR (ou exclusif)	`$res = $a xor $b;`	$a ou $b (exclusivement) retourne TRUE
! (faux)	`$res = ! $a;`	$a retourne FALSE

(*) priorité inférieure à celle de l'opérateur précédent[17]

4.10 Outils utiles

var_dump() est une procédure qui affiche le contenu d'une variable avec des informations de type[18].

```php
$a = 1.234;
var_dump($a);
$a = array("a", "bb", "ccc");
var_dump($a);
```

```
float 1.234

array (size=3)
  0 => string 'a' (length=1)
  1 => string 'bb' (length=2)
  2 => string 'ccc' (length=3)
```

16 http://php.net/manual/fr/language.operators.logical.php
17 http://php.net/manual/fr/language.operators.precedence.php
18 http://php.net/manual/fr/function.var-dump.php

print_r() est une procédure qui affiche le contenu d'une variable sans information de type[19]

```php
$a = array('a', 'bb', 'ccc');
echo '<pre>';
print_r($a);
echo '</pre>';
```

```
Array
(
    [0] => a
    [1] => bb
    [2] => ccc
)
```

19 http://php.net/manual/fr/function.print-r.php

5 Tableaux

Un tableau est une structure de donnée qui permet de rassembler plusieurs données sous un même nom. PHP offre différentes sortes de tableaux[20]

5.1 Tableau indicé

Une manière simple de déclarer un tableau est d'utiliser la fonction array() :

```
<?php
 $tablo = array ('premier', 'deuxième','troisième');
?>
```

Un tableau peut aussi être déclaré et initialisé la syntaxe courte depuis PHP 5.4

```
<?php
 $tablo2=[10,20,30];
?>
```

La fonction `print_r` permet d'afficher une variable, quel que soit son type :

```
<?php print_r ($tablo); ?>
```

retourne :

```
Array (
[0] => premier
[1] => deuxième
[2] => troisième
)
```

Ceci montre la structure ([indice] => élément) du tableau. Ajoutons un élément à notre tableau :

```
<?php
 $tablo[] = 'nouveau';
 print_r($tablo);
?>
```

retourne :

```
Array (
  [0] => premier
  [1] => deuxième
  [2] => troisième
  [3] => nouveau
 )
```

20 http://php.net/manual/fr/language.types.array.php

5.2 Tableau associatif

On parle de tableaux associatifs lorsque l'on affecte une valeur de type chaîne de caractères à l'indice en lieu et place d'un entier. On parle alors de "clé" plutôt que "d'indice".

Initiation

```
<?php
 $tablo = array ('patron' => 'Marc-André',
                 'commercial' => 'Louis-Christophe',
                 'Thierry'
                 );
 print_r ($tablo);
?>
```

retourne :

```
Array (
   [patron] => Marc-André
   [commercial] => Louis-Christophe
   [0] => Thierry
  )
```

L'indice [0] est automatiquement affecté au premier élément dépourvu de clé. Cela n'a pas beaucoup de sens.

Suppression d'un élément :

La fonction unset() permet de supprimer une variable. Il est aussi possible de l'appliquer uniquement à une cellule du tableau :

```
<?php
 unset ($tablo[0]);
 print_r($tablo);
?>
```

retourne :

```
 Array (
 [patron] => Marc-André
 [commercial] => Louis-Christophe
 )
```

Ajout d'un élément avec un indice associé :

```
<?php $tablo['secrétaire'] = 'Valérie-Anne'; ?>
```

Affichage d'un élément d'après son indice :

```
<?php echo $tablo['secrétaire']; ?>
```

retourne : `Valérie-Anne`

5.3 Tableau multidimensionnels

On peut considérer un tableau multidimensionnel comme un tableau de tableau. Sa déclaration en syntaxe courte correspond à ceci :

```php
$car_one=["Volvo", 22, 18];
$car_two=["BMW",15,13];
$cars3=[
    "auto 1"=>$car_one,
    "auto 2"=>$car_two
];
var_dump($cars3);
```

```
array (size=2)
  'auto 1' =>
    array (size=3)
      0 => string 'Volvo' (length=5)
      1 => int 22
      2 => int 18
  'auto 2' =>
    array (size=3)
      0 => string 'BMW' (length=3)
      1 => int 15
      2 => int 13
```

D'autre déclarations sont possibles :

```php
$cars = array(
    array("Volvo", 22, 18),
    array("BMW",15,13),
    array("Saab",5,2),
    array("Land Rover",17,18)
);


$cars2=[
    "auto 1"=>["Volvo", 22, 18],
    "auto 2"=>["BMW",15,13],
    "auto 3"=>["Saab",5,2],
    "auto 4"=>["Land Rover",17,18]
];
```

```
array (size=4)
  0 =>
    array (size=3)
      0 => string 'Volvo' (length=5)
      1 => int 22
      2 => int 18
  1 =>
    array (size=3)
      0 => string 'BMW' (length=3)
      1 => int 15
      2 => int 13
  2 =>
    array (size=3)
      0 => string 'Saab' (length=4)
      1 => int 5
      2 => int 2
  3 =>
    array (size=3)
      0 => string 'Land Rover' (length=10)
      1 => int 17
      2 => int 18
```

Accès à une case du tableau avec la syntaxe des crochets :

```php
//lecture et écriture
echo $cars2["auto 1"][0];
```

retourne : `Volvo`

```php
$cars2["auto 1"][0]="Mazda";
echo $cars2["auto 1"][0];
```

retourne : `Mazda`

Ajout d'une ligne au tableau :

```php
$cars2["auto 5"]=["Peugeot", 54,36];
```

5.4 Parcourir un tableau

Trois méthodes :

- La boucle for
- La boucle foreach
- Les fonctions print_r et var_dump (utilisée principalement en phase de développement)

A l'aide d'une boucle for

Utile si on connaît le nombre d'éléments du tableau et si l'indice est numérique :

```
<?php
       // création d'un tableau
          $peintres = array ('Picasso', 'Van Gogh', 'Rothko');

          // affichage du contenu
          $nbEle = count($peintres);
          for ($indice = 0; $indice < $nbEle; $indice++) {
             echo $peintres[$indice],'<br/>';
          }
?>
```

A l'aide d'une boucle foreach

Cette boucle permet de parcourir une collection sans connaître préalablement le nombre d'élément qu'elle contient ni la valeur des clés. Elle est très bien adaptée au parcours de tableaux.

```
<?php
       // création d'un tableau
       $peintres = array ('Picasso', 'Van Gogh', 'Rothko');

       // affichage du contenu
       foreach ($peintres as $unPeintre) {
          echo $unPeintre,'<br/>';
       }

       foreach ($peintres as $indice => $unPeintre) {
           echo $indice, ' - ', $unPeintre,'<br/>';
       }
?>
```

La dernière syntaxe permet de récupérer aussi bien la valeur contenue dans la cellule que le numéro de la cellule. Cette approche est particulièrement utiles avec les tableau associatifs.

Parcours d'un tableau multidimensionnel

Le parcours se réalise avec une boucle imbriquée pour chaque dimension. Pour un tableau à deux dimensions on obtient ceci :

```php
    $garage=[
        "JU2296"=>["Volvo", 22, 18],
        "NE6789"=>["BMW",15,13],
        "GR4379"=>["Saab",5,2],
        "VS9666"=>["Land Rover",17,18]
    ];

foreach ($garage as $imatriculation => $voiture){
   echo '<h4>',$imatriculation, ":",'</h4>';
       foreach($voiture as $carctersitique => $valeur)
           echo '<p>',$carctersitique, ': ',$valeur, '</p>';
   }
```

Dans cet exemple, la première boucle extrait les voitures du garage alors que la seconde boucle parcours les caractéristiques de chaque voiture extraite

5.5 Quelques fonctions de manipulation de tableau

Vérifier si une clé existe

La fonction booléenne `array_key_exists` retourne vrai si la clé se trouve dans le tableau, faux sinon :

```
array_key_exists ('cle', $array)
```

```php
<?php
    $tablo = array ('patron' => 'Marc-André',
       'commercial' => 'Louis-Christophe',
       'financier' => 'Jeanne'
    );

    if (array_key_exists ('financier', $tablo)) {
       echo 'La clé "financier" existe dans le tableau';
    }
?>
```

Vérifier si une valeur existe

La fonction booléenne `in_array` retourne vrai si la valeur se trouve dans le tableau, faux sinon.

```php
<?php
   $peintres = array ('Picasso', 'Van Gogh', 'Rothko');

   if (in_array ('Rothko', $peintres)) {
       echo 'Le peintre "Rothko" existe dans le tableau', '<br />';
   }
?>
```

Récupérer la clé associée à une valeur

La fonction `array_search` retourne la clé correspondant à la valeur (le numéro de l'indice si c'est un array et le nom de la clé si c'est un tableau associatif). Si la valeur n'existe pas dans le tableau, la fonction retourne faux.

```
<?php
```

```
    // On admet que les tableaux sont déclarés
    $pos = array_search ('Rothko', $peintres);
    echo '"Rothko" se trouve à l\'indice ', $pos, '<br />'; // 2

    $pos = array_search ('Jeanne', $tablo);
    echo '"Jeanne" se trouve à la clé ', $pos, '<br />';  // financier
?>
```

Appliquer un traitement par lot

La fonction `array_map`[21] applique un traitement standard ou personnalisé à tous les éléments d'un tableau.

```
$a =[
   'nom' => 'Monnet ',
   'prenom' => ' Paul '
];
var_dump($a);
$b=array_map('trim',$a);
var_dump($b);
```

Avant l'utilisation de `array_map` :	Après l'utilisation de `array_map` :

`array (size=2)`
`'nom' => string 'Monnet ' (length=7)`
`'prenom' => string ' Paul ' (length=6)`
	`array (size=2)`
	`'nom' => string 'Monnet' (length=6)`
	`'prenom' => string 'Paul' (length=4)`

L'appel de la fonction `trim`[22] enlève les espaces avant et après la chaîne de caractères

Le traitement peut appeler un fonction personnalisé

```
function cube($n)
{
    return($n * $n * $n);
}
$a = array(1, 2, 3, 4, 5);
$b = array_map("cube", $a);
var_dump($b);
```

Le traitement de la fonction cube sera appliquer sur chaque élément du tableau.

5.6 Variables superglobales

Il est fréquent d'avoir besoin d'informations en rapport avec le contexte d'exécution de PHP. Dans un environnement web, ce peut être le nom du serveur, savoir d'où vient le visiteur ou récupérer l'adresse IP du client.

Pour la récupération des informations envoyés par un formulaire HTML, nous utiliserons `$_GET` et `$_POST` qui sont aussi des variables superglobales.

21 http://php.net/manual/fr/function.array-map.php
22 http://php.net/manual/fr/function.trim.php

Plusieurs variables prédéfinies en PHP sont *superglobales*[23], ce qui signifie qu'elles sont disponibles quel que soit le contexte du script.

$_SERVER

C'est un tableau associatif contenant des informations comme les en-têtes, dossiers et chemins du script. Les entrées de ce tableau sont créées par le serveur web. Quelques clés de ce tableau :

`SERVER_NAME`

Le nom relatif du serveur qui exécute le script courant :

```php
<?php echo $_SERVER['SERVER_NAME']; ?>
```

Retour possible: `localhost`

`DOCUMENT_ROOT`

Le chemin du répertoire racine du serveur.

```php
<?php echo $_SERVER['DOCUMENT_ROOT']; ?>
```

Retour possible: `C:/wamp/www`

`PHP_SELF`

Le nom du fichier du script courant, depuis la racine ($_SERVER['DOCUMENT_ROOT']) du serveur:

```php
<?php echo $_SERVER['PHP_SELF']; ?>
```

Retour possible : `/demo/test_$_SERVER.php`

`HTTP_USER_AGENT`

Décrit le client de la requête (votre navigateur) :

```php
<?php echo $_SERVER['HTTP_USER_AGENT']; ?>
```

Retours possibles :

```
Mozilla/5.0 (compatible, MSIE 11, Windows NT 6.3; Trident/7.0;
rv:11.0) like Gecko ternet Explorer 11.0
```

23 http://php.net/language.variables.superglobals

6 Fonctions

Une fonction est une série d'instructions qui effectuent des actions. Une fonction peut retourner une valeur. Lorsqu'une fonction ne retourne rien on parle alors de procédure.

Une fonction se définit par le mot clé `function` suivi du nom de la fonction, d'une liste de paramètres entre parenthèses et du corps de la fonction entre accolades.

Une fonction peut être définie n'importe où dans la page. Il n'est pas nécessaire qu'elle soit définie avant son appel. Il est d'usage de les regrouper en début de code.

Il est également possible voire conseillé, d'écrire les fonctions dans un fichier séparé et de l'inclure dans le programme nécessitant leur utilisation.

```
<?php
function affiche() {
    echo 'Hello world!', '<br>';
}
?>
<!doctype html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
          content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-
scale=1.0, minimum-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Document</title>
</head>
<body>
<?php
    affiche();// Appel de la fonction.
?>
</body>
</html>
```

6.1 Portée des variables

Locale : Accès de l'intérieur d'une fonction.

Globale : Accès de l'extérieur de toute fonction.

Les variables globales sont accessibles dans les fonctions avec la déclaration suivante[24] :

```
global $nomVariable;
```

```
function testVarGlobale(){
    global $gVar;
    $gVar = 17;
    $GLOBALS['gVar'] = 30;// Syntaxe alternative,
}
// Initialisation de la variable globale.
$gVar = 11;
echo 'debut programme ', $gVar, '<br />'; // 11.
testVarGlobale();
echo 'fin programme ', $gVar, '<br />'; // 30.
```

24 http://php.net/manual/fr/language.variables.scope.php

6.2 Paramètres

La liste de paramètres désigne une liste de variables locales initialisées par les valeurs des paramètres effectifs lors de l'appel.

Vocabulaire :

Paramètre formel : variables entre parenthèses, dans l'en-tête de la fonction

Paramètre effectif : valeurs ou variables passées à la fonction lors de son appel

Paramètres en entrée seulement

Le mécanisme de passage est un passage par valeur. La fonction reçoit une valeur qu'elle stocke dans son paramètre formel correspondant. Le paramètre effectif n'est pas modifié :

```php
<?php
function test ($p){
    echo 'debut test', $p, '<br />';
    $p = 99;
    echo 'fin test', $p, '<br />';
}
$var = 11;

// Affiche : 11.
echo 'debut programme ', $var, '<br />';

// Affiche 11.
test ($var); // Affiche 11 puis 99

// Affiche : 11.
echo 'fin programme ', $var, '<br />';
?>
```

```
debut programme: 11
debut test: 11
fin test: 99
fin programme: 11
```

```
debut programme: 11
debut test: 11
fin test: 99
fin programme: 99
```

Paramètres en entrée / sortie

Pour passer des paramètres en entrée-sortie il faudra utiliser le passage par référence. Il faut préfixer le nom du paramètre par le symbole **&**.

```php
<?php
function testRef (&$p){
    echo 'debut test: ', $p, '<br />';
    $p = 99;
    echo 'fin test: ', $p, '<br />';
}
$var = 11;
// Affiche : 11.
echo 'debut programme: ', $var, '<br />';
// Affiche 11.
testRef($var); // Affiche 11 puis 99
echo 'fin programme: ', $var, '<br />';
?>
```

6.3 Retour d'une fonction

Si la fonction retourne une valeur on utilisera l'instruction `return [expression]`. Cette instruction interrompt le code de la fonction et retourne la valeur de l'expression à l'appelant.

```php
<?php
function multiplie ($nb1, $nb2) {
    $resultat = ($nb1 * $nb2);
    // On peut aussi écrire directement return $nb1 *$nb2.
    return $resultat;
}
// Exemple d'appel.
$val1 = 15;
$val2 = 3;
// 15* 3 = 45.
echo $val1, ' * ', $val2, ' = ', multiplie ($val1, $val2);
?>
```

7 Formatage du code PHP

Chaque programmeur a ses propres sensibilités quant à l'écriture du code PHP. Pour exprimer un numéro de téléphone, certains choisirons :

- numero_telephone (notation dite *snake case notation*[25]),
- NumeroTelephone (notation dite upper camel case notation[26])
- numeroTelephone (notation dite lower camel case notation)
- …

Lorsque plusieurs développeurs interviennent dans un projet on souhaite toutefois obtenir une certaine homogénéité. Les meilleures pratiques issus des entreprises sont regroupées dans des catalogues de prescriptions. Les grands acteurs du domaine, notamment Zend[27] proposent de telles directives. Il faut aussi compter avec les standards PHP établis par le *Framework Interoperability Group*[28]. Plus globalement pour le CEJEF, Steve Fallet a regroupé les prescriptions importantes dans notre contexte[29].

7.1 Formatage des fichiers PHP

Général

Pour les fichiers contenant uniquement du code PHP, la balise de fermeture ("?>") n'est jamais permise. Elle n'est pas requise par PHP. Ne pas l'inclure permet de se prémunir des problèmes liés à l'injection accidentelle d'espaces blancs dans la sortie.

Indentation

Utilisez une indentation de 4 espaces, sans tabulations.

Longueur maximum d'une ligne

La longueur souhaitée d'une ligne est de 80 caractères, c'est-à-dire que les développeurs devraient avoir pour but de ne pas dépasser les 80 caractères pour des raisons pratiques. Cependant, des lignes plus longues sont acceptables.

La longueur maximum de toute ligne de code PHP est de 120 caractères.

Terminaison de lignes

La terminaison de ligne est la terminaison standard pour les fichiers textes UNIX. Les lignes doivent finir seulement avec un "linefeed" (LF). Les linefeeds sont représentés comme 10 en

25 https://fr.wikipedia.org/wiki/Snake_case
26 https://fr.wikipedia.org/wiki/CamelCase
27 https://framework.zend.com/manual/1.10/en/coding-standard.naming-conventions.html
28 http://www.php-fig.org/psr/
29 [SFA15], https://github.com/fallinov/151A/wiki

ordinal, ou 0x0A en hexadécimal.

Note : N'utilisez pas de retour chariots (CR) comme le font les Macintosh (0x0D) ou de combinaison retour chariot/linefeed (CRLF) comme le font les ordinateurs sous Windows (0x0D, 0x0A).

7.2 Conventions de nommage

Noms de fichiers

Pour tous les fichiers, seuls des caractères alphanumériques, tirets bas et tirets demi-cadratin ("-") sont autorisés. Les espaces et les caractères spéciaux sont interdits.

Note : L'utilisation de la "notationCamel" est réservée aux fichier décrivant des Class, objets.

Tout fichier contenant du code PHP doit se terminer par l'extension ".php". Ces exemples montrent des noms de fichiers acceptables :

```
personnes.php
personnes_ajout.php
personnes-ajout.php
personnesAjout.php
```

Fonctions et méthodes

Les noms de fonctions ne peuvent contenir que des caractères alphanumériques. Les tirets bas ("_") ne sont pas permis. Les nombres sont autorisés mais déconseillés.

Les noms de fonctions doivent toujours commencer avec une lettre en minuscule. Quand un nom de fonction est composé de plus d'un seul mot, la première lettre de chaque mot doit être mise en majuscule. C'est ce que l'on appelle communément la "notationCamel".

La clarté est conseillée. Le nom des fonctions devrait être aussi explicite que possible, c'est un gage de compréhension du code.

Voici des exemples de noms acceptables pour des fonctions :

```
validerFormulaire()
afficherProduit()
supprimerPersonne()
```

Variables

Les noms de variables ne peuvent contenir que des caractères alphanumériques. Les tirets bas ne sont pas permis. Les nombres sont autorisés mais déconseillés.

Tout comme les noms de fonction (cf. la section ci-dessus), les noms de variables doivent toujours commencer par un caractère en minuscule et suivre la convention de capitalisation de la "notationCamel".

La clarté est conseillée. Les variables devraient toujours être aussi claires que pratiques. Des

noms de variables comme "$i" et "$n" sont déconseillé pour tout autre usage que les petites boucles. Si une boucle contient plus de 20 lignes de code, les variables pour les indices doivent avoir des noms descriptifs.

Constantes

Les constantes peuvent contenir des caractères alphanumériques et des tirets bas. Les nombres sont autorisés.

Les constantes doivent toujours être en majuscule, cependant les mots pouvant les composer doivent être séparés par des tiret-bats ("_").

Par exemple, UTILISATEUR_NOM est permis mais UTILISATEURNOM ne l'est pas.

7.3 Style de codage

7.3.1 Démarcation du code PHP

Les codes PHP doivent toujours être délimités dans la forme complète, par les balises PHP standards :

```
<?php    ?>
```

Les balises courtes d'ouvertures ("<?") ne sont pas autorisées. Pour les fichiers ne contenant que du code PHP, la balise de fermeture doit toujours être omise (Voir le chapitre 7.1 Formatage des fichiers PHP).

7.3.2 Chaîne de caractères

Chaînes littérales

Lorsqu'une chaîne est littérale (c'est-à-dire qu'elle ne contient pas de substitution de variables), l'apostrophe ou guillemet simple doit être utilisé pour démarquer la chaîne.

```
$a = 'Exemple de chaîne de caractères';
```

Chaînes de caractères littérales avec apostrophes

Lorsque qu'une chaîne littérale contient des apostrophes, il est permis de les démarquer en utilisant les guillemets doubles. Ceci est particulièrement conseillé pour les requêtes SQL

```
$sql = "SELECT id, name from people "
     . "WHERE name='Eric' OR name='Caroline'";
```

La syntaxe ci-dessus est préférée à l'échappement des apostrophes car elle est plus facile à lire.

Syntaxe Heredocs

Une autre façon de délimiter une chaîne de caractères est la syntaxe Heredoc. Elle est pratique et plus facile à lire lorsqu'on doit délimiter des chaînes sur plusieurs lignes, comme

les requêtes SQL.

```
$sql = <<<EOT
SELECT id,
       name
FROM   people
WHERE  name = 'Eric'
       OR name = 'Caroline'
EOT;
```

Substitution de variables

La substitution des variables est permise en utilisant une de ces deux formes :

```
$greeting = "Bonjour $name, bienvenue !";

$greeting = "Bonjour {$name}, bienvenue !";
```

La seconde méthode ne s'utilise pratiquement plus.

Concaténation de chaînes

Les chaînes peuvent êtres concaténées en utilisant l'opérateur "**.**". Un espace doit toujours être ajouté avant, et après cet opérateur, cela permet d'améliorer la lisibilité :

```
$company = 'Zend' . ' ' . 'Technologies';
```

Lors de la concaténation de chaînes avec l'opérateur ".", il est permis de couper le segment en plusieurs lignes pour améliorer la lisibilité. Dans ces cas, chaque nouvelle ligne doit être remplie avec des espaces, de façon à aligner le "." sous l'opérateur "=" :

```
$sql = "SELECT `id`, `name` FROM `people` "
     . "WHERE `name` = 'Caroline' "
     . "ORDER BY `name` ASC ";
```

7.3.3 Tableaux

Tableaux indexés numériquement

L'utilisation d'indices négatifs n'est pas permise.

Un tableau indexé peut commencer avec n'importe quel nombre positif, cependant cette méthode est déconseillée. Il est conseillé de commencer l'indexation à 0.

Lors de la déclaration de tableaux indexés avec la construction array, un espace doit être ajouté après chaque virgule délimitante, pour améliorer la lisibilité :

```
$sampleArray = array(1, 2, 3, 'Jean', 'Némarre');
```

Il est aussi permis de déclarer des tableaux indexés sur plusieurs lignes en utilisant la construction array. Dans ce cas, chaque nouvelle ligne doit être remplie par des espaces jusqu'à ce que cette ligne s'aligne, comme il est montré dans l'exemple suivant :

```
$sampleArray = array(1, 2, 3, 'Jean', 'Némarre',
                     $a, $b, $c,
```

```
                      56.44, $d, 500);
```

Autre solutions, commencer sur la première ligne et indenter les lignes suivantes d'un niveau puis terminer la déclaration au même niveau avec ");".

```
$sampleArray = array(1, 2, 3, 'Jean', 'Némarre',
    $a, $b, $c,
    56.44, $d, 500,
);
```

Lorsque qu'on utilise cette déclaration, nous encourageons l'utilisation d'une "virgule de fuite" pour le dernier élément du tableau. Cela facilite l'ajout de nouvelles valeurs et évites les erreurs d'oubli de virgule.

Tableaux associatifs

Lorsque de la déclaration de tableaux associatifs avec la construction array, il est conseillé de séparer la définition sur plusieurs lignes. Dans ce cas, chaque ligne successive doit être remplie par des espaces pour que les clés et les valeurs soient alignées :

```
$sampleArray = array('firstKey'  => 'firstValue',
                     'secondKey' => 'secondValue');
```

Alternative avec "virgule de fuite" :

```
$sampleArray = array(
    'firstKey'  => 'firstValue',
    'secondKey' => 'secondValue',
);
```

7.4 Fonctions et méthodes

Déclaration de fonctions et de méthodes

L'accolade ouvrante est toujours écrite après la parenthèse fermante des arguments. Il n'y a pas d'espace entre le nom de la fonction et les parenthèses des arguments. Il a un espace entre la parenthèse fermante et l'accolade.

Voici un exemple d'une déclaration permise d'une fonction de classe :

```
/**
* Bloc de documentation
*/
function bar() {
    // contenu de la fonction
    // qui doit être indenté avec 4 espaces
}
```

Au cas ou la liste des arguments dépasse le maximum de caractères autorisés par ligne, vous devez ajouter un saut de ligne. Les arguments additionnels seront simplement indenté d'une

unité pour chaque ligne suivante.

Il faudra ajouté un retour après le dernier argument, fermer la parenthèse, ajouter un espace et ouvrir une accolade sur la même ligne.

```
/**
* Documentation Block Here
*/
function bar($arg1, $arg2, $arg3,
    $arg4, $arg5, $arg6
) {
    // all contents of function
    // must be indented four spaces
}
```

Le passage par référence est permis uniquement dans la déclaration de la fonction :

```
/**
* Bloc de documentation
*/
function bar(&$baz) {
    // all contents of function
    // must be indented four spaces
}
```

L'appel par référence est interdit.

La valeur de retour ne doit pas être entourée de parenthèses. Ceci peut gêner à la lecture et peut aussi casser le code si une méthode est modifiée plus tard pour retourner par référence.

```
/**
* INCORRECT
*/
function bar() {
    return($this->bar);
}
```

```
/**
* CORRECT
*/
function bar() {
    return $this->bar;
}
```

7.4.1 Usage de fonctions et méthodes

Les arguments d'une fonction sont séparés par un espace après la virgule de délimitation. Voici un exemple d'un appel de fonction qui prend trois arguments :

```
threeArguments(1, 2, 3);
```

L'appel par référence est interdit. Référez vous à la section sur la déclaration de fonctions pour la méthode correcte de passage des argument par référence.

Pour les fonctions dont les arguments peuvent être des tableaux, l'appel à la fonction doit inclure la construction "`array`" et peut être divisé en plusieurs ligne pour améliorer la lecture. Dans ces cas, les standards d'écriture de tableaux s'appliquent aussi :

```
threeArguments(array(1, 2, 3), 2, 3);
```

```
threeArguments(array(1, 2, 3, 'Zend', 'Studio',
                     $a, $b, $c,
                     56.44, $d, 500), 2, 3);
```

```
threeArguments(array(
    1, 2, 3, 'Zend', 'Studio',
    $a, $b, $c,
    56.44, $d, 500
), 2, 3);
```

7.5 Structure de contrôle

If / Else / Elseif

Les structures de contrôles basées sur les constructions if et elseif doivent avoir un seul espace avant la parenthèse ouvrante de la condition, et un seul espace après la parenthèse fermante.

Pour la condition entre les parenthèses, les opérateurs doivent être séparés par des espaces pour une meilleure lisibilité. Les parenthèses internes sont conseillées pour améliorer le regroupement logique de longues conditions.

L'accolade ouvrante est écrite sur la même ligne que la condition. L'accolade fermante est toujours écrite sur sa propre ligne. Tout contenu présent à l'intérieur des accolades doit être indenté par 4 espaces.

```
if ($a != 2) {
    $a = 2;
}
```

Si la condition dépasse le maximum de caractères autorisés par ligne, on ajoutera un retour avant chaque opérateur logique.

Dans ce cas, chaque ligne successive doit être remplie par des espaces pour que opérateurs logiques soient alignés. On fermera la condition sur une nouvelle ligne au même niveau que le début de la condition, suivi d'un espace et de l'accolade d'ouverture.

```
if (($a == $b)
    && ($b == $c)
    || (Foo::CONST == $d)
) {
    $a = $d;
}
```

Pour les instructions if qui incluent elseif ou else, les conventions de formatage sont similaires à celles de la construction if. Les exemples suivants montrent le formatage approprié pour les structures if avec else et/ou les constructions elseif :

```
if ($a != 2) {
    $a = 2;
} else {
    $a = 7;
}
```

```
if ($a != 2) {
    $a = 2;
} elseif ($a == 3) {
    $a = 4;
} else {
    $a = 7;
}
```

PHP permet que ces instructions soient écrites sans accolades dans certaines circonstances.

La convention de codage ne fait pas de différentiation et toutes les instructions if, elseif et else doivent utiliser des accolades.

Switch

Les instructions de contrôle avec switch ne doivent avoir qu'un seul espace avant la parenthèse ouvrante de l'instruction conditionnelle, et aussi un seul espace après la parenthèse fermante.

Tout le contenu à l'intérieur de l'instruction switch doit être indenté avec 4 espaces. Le contenu sous chaque "case" doit être indenté avec encore 4 espaces supplémentaires.

```
switch ($numPeople) {
    case 1:
        break;
```

```
    case 2:
        break;

    default:
        break;
}
```

La construction default ne doit jamais être oubliée dans une instruction switch.

Note : Il est parfois pratique d'écrire une clause case qui passe à travers le case suivant en omettant l'inclusion de break ou return. Pour distinguer ce cas d'un bug, toute clause case ne contenant pas break ou return doit contenir le commentaire `// break intentionally omitted`

7.6 Documentation

Le code lui-même fait partie de la documentation. Il existe des règles de syntaxe pour la documentation intégrée au code.

Format de la documentation

Tous les blocs de documentation ("docblocks") doivent être compatible avec le format phpDocumentor. La description du format phpDocumentor n'est pas du ressort de ce document. Pour plus d'information, visitez » http://phpdoc.org/

Tous les fichiers de code source doivent contenir un docblock du fichier, en haut de chaque fichier.

Fichiers

Chaque fichier qui contient du code PHP doit avoir un bloc d'entête en haut du fichier qui contient au minimum ces balises phpDocumentor :

```
/**
* Description courte du fichier
*
* Description longue du fichier s'il y en a une
*
* @author Jean Némarre <j.nemarre@divtec.ch>
* @copyright  Copyright (c) 2015 DIVTEC, Porrentruy, CH (http://www.divtec.ch)
* @version    1.0.1
*/
```

Version automatique :

```
* @version    $Id:$
```

Fonctions

Chaque fonction, méthode, doit avoir un docblock contenant au minimum :

Une description de la fonction

Tous les arguments

Toutes les valeurs de retour possibles

```
/**
* Retourne le nom et l'âge passés en paramètres
*
* @param    string        nom de la personne
* @param    integer       age de la personne
* @return   string|null   Chaine contenant le nom et l'âge de la personne ou null si pas d'age
*/
function PersonneNomAge($nom, $age)
{
    if ($age) {
        return $nom . " " . $age . " ans";
    }
    return null;
}
```

8 Transmission d'informations et inclusions de fichiers

Une application web contient plusieurs pages reliées entre elles par des liens ou des redirections automatiques. En outre, les pages et fichiers de l'application se partagent des données, souvent propres à la session de navigation d'un internaute (cas du panier d'achats)

Il existe différentes techniques pour réaliser ces transferts d'informations et ces liens entre les pages :

Navigation entre les pages et fichiers de l'application :

- Au travers de liens hypertexte
- Au travers de redirection instantanée ou temporisée

Transfert d'informations entre les pages :

- Au travers de l'envoi de formulaire avec les méthodes POST et GET
- Au travers de liens paramétré
- Au travers de redirections paramétrées
- Au travers de cookies sur le navigateur
- Au travers de variables de session sur le serveur

Ce chapitre passe en revue ces différentes techniques sans toutefois revenir sur la gestion de formulaires

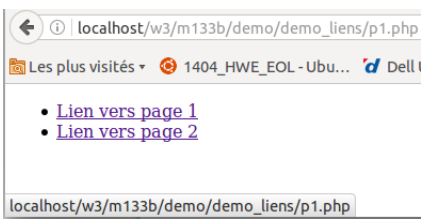
8.1 Navigation entre les pages et fichiers de l'application

La navigation par liens constitue la base même du web. Les liens peuvent être paramétrés afin d'envoyer des informations entre les page du site.

8.1.1 Liens hypertextes

Les liens sont des éléments HTML cliquables dont la syntaxe est la suivante :

```
<ul>
    <li><a href="p1.php">Lien vers page 1</a></li>
    <li><a href="p2.php">Lien vers page 2</a></li>
</ul>
```



8.1.2 Lien paramétrés

Un lien paramétré est cliquable et permet simplement de transmettre une information via la méthode GET. La syntaxe de l'URL générée correspond à celle des formulaires envoyés avec la méthode GET. Le paramètre doit être ajouter à la suite de l'url

```
<ul>
    <li><a href="p1.php">Lien vers page 1</a></li>
    <li><a href="p2.php?id=123">Lien paramétré vers page 2</a></li>
</ul>
```

Pour ajouter plusieurs paramètres, il faut les séparer par le caractère &.

L'ajout du paramètre n'est pas forcément écrit en dur dans le code. Normalement, il est issus du code PHP du script.

```
<ul>
    <li><a href="p1.php?num= <?php echo $num; ?>">Lien paramétrée vers page 1</a></li>
    <li><a href="p2.php">Lien vers page 2</a></li>
</ul>
```

Récupération des données du lien

La récupération du paramètre dans la page cible se réalise en lisant le contenu de la variable superglobale `$_GET` . Une clé nommée comme le paramètre passé dans l'URL doit exister. Avant de l'utiliser, il faut vérifier que la variable existe et soit utilisable. Dans le cas du passage d'un nombre . Le code de la page cible peut ressembler à ceci :

```
<?php

$error = '';
// Renvoie FAUX lorsque ce n'est pas un nombre
$num = filter_input(INPUT_GET, 'num', FILTER_VALIDATE_INT);

// Traite la réponse du filtre.
// Vérifie que ce soit un nombre ou du moins que la variable n'est pas vide.
if (($num === false) || ($num === null)) {
    $error = 'pas de paramètre valide';
    $num = '';
}
```

Le filtre de validation peut être paramétré pour réagir à une plage donnée[30].

8.2 Redirections

Il existe des applications web pour lesquelles on souhaite rediriger le visiteur en fonction de

30 http://php.net/manual/fr/function.filter-input.php

paramètres. C'est le cas par exemple pour un script d'identification. Si l'internaute fournit les bons identifiants alors il est redirigé automatiquement vers son espace personnel, sinon il est renvoyé vers le formulaire d'authentification.[31]

Lorsque l'on souhaite créer une redirection avec PHP, on utilise une fonction permettant d'envoyer des entêtes de type *location* (adresse). Pour cela, PHP dispose de la fonction `header()` qui se charge d'envoyer les entêtes passés en paramètre.

> **Règle importante** : l'appel de cette fonction doit se faire avant tout envoi au navigateur (instruction echo, balise html...) sous peine de générer une erreur de type Headers already sent by.... Cette erreur signifie que la page a déjà été envoyée au navigateur avant de vouloir envoyer des entêtes HTTP. La logique de développement demande le contraire !

Fonction header

La syntaxe de la redirection est simple. Elle reçoit en paramètre une chaîne de caractères précisant le type d'entête (location) et la page sur laquelle l'internaute doit être renvoyé.

Syntaxe :

```
void header ( string $string [, bool $replace = true [, int $http_response_code ]] )
```

- Le script ne doit rien envoyer avant cette instruction. En effet, nous sommes en train d'envoyer un en-tête HTTP.
- Le script continue à s'exécuter (les instructions qui suivent l'appel seront exécutées). Il est de ce fait conseillé de mettre un exit après `header(…)` avec l'appel de la fonction `exit`.

Exemples :

1. Redirection simple :

```
<?php
    header("location: p1.php");
    exit;
?>
```

2. Redirection temporisée

```
<body>
<p>Redirection</p>
<?php
    header('refresh:5; url=p1.php');
    echo 'Dans 5 secondes vous serez redirigé vers la page1...';
    //exit;
?>
<p> Message après redirection visible s'il n'y a pas d'EXIT</p>
</body>
```

31 Selon [PHP-07]

3. Redirection paramétrée

Au même titre qu'un lien, une redirection peut recevoir des paramètres à transmettre selon la méthode GET :

```
<?php
        header("Location: page1.php?prenom=Albert&nom=leVert");
exit;
?>
```

8.3 Inclusion de fichiers

PHP offre deux fonctions simples pour réutiliser du code ou simplement inclure des fragments de pages HTML. Ces possibilités permettent de créer des fichiers communs à plusieurs scripts PHP. L'extension de ces fichiers est généralement .inc ou .inc.php

Exemples d'utilisation :

- Insérer un en-tête de page ou un menu identique sur plusieurs pages d'un site.
- Rassembler des fonctions PHP utilisées dans un même fichier pour le rendre réutilisable.

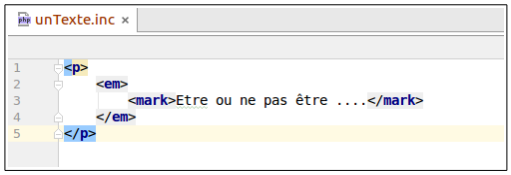
Fonction Include

Le contenu du fichier sera inclus dynamiquement lors de l'exécution du code. L'instruction est réévaluée à chaque passage et ne provoque qu'une alarme en cas d'erreur[32]. Le traitement continue.

L'instruction `include` est utilisées pour inclure des fragments de page HTML.Par exemple, un menu, un en-tête ou un pied de page que l'on souhaite identique sur toutes les pages d'un site.

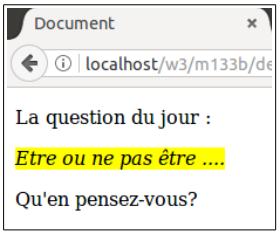
Fonctionnement à travers un exemple simple :

Contenu du fichier de fragment HTML



Contenu du fichier l'utilisant et résultat

```
<body>
    <p>La question du jour :</p>
    <?php include('unTexte.inc'); ?>
    <p>Qu'en pensez-vous?</p>
</body>
```



32 cf. http://php.net/manual/fr/function.include.php

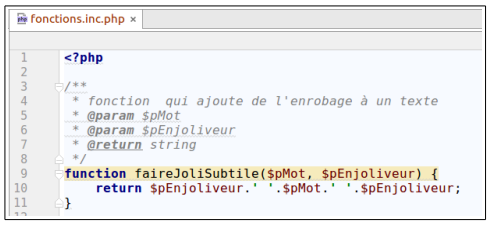
Fonction Require

Cette instruction provoque une erreur fatale en cas d'échec[33]. De ce fait, le traitement prévu par le script s'interrompt.

La fonction `require_once()` est une alternative qui s'assure que le fichier que l'on essaie d'inclure ne l'a pas déjà été. Ce fonctionnement est utile s'il s'agit d'un fichier de fonctions PHP car permet d'éviter les déclarations multiples qui engendrent des erreurs.

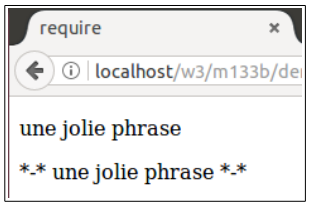
Pour l'inclusion de fichiers de fonctions PHP il est courant d'utiliser la fonction `require_once()`.

Contenu du fichier de fonction



Contenu du fichier l'utilisant et résultat

```php
<?php
require_once('./fonctions.inc.php');
$phrase = 'une jolie phrase';
define('MOTIF', '*-*');
?>
<!DOCTYPE html >
<html lang="fr">
<head>
    <meta charset="utf-8"/>
    <title>require</title>
</head>
<body>
    <p> <?php echo $phrase; ?> </p>
    <p> <?php echo faireJoliSubtile($phrase, MOTIF); ?> </p>
</body>
</html>
```



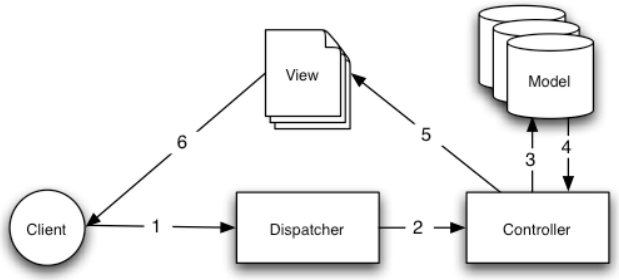
33 cf. http://php.net/manual/fr/function.require.php

9 Architecture MVC simplifiée d'une application web

Le Modèle-Vue-Contrôleur[34] (abr. MVC) est un motif d'architecture logicielle destiné aux interfaces graphiques très populaire pour les applications web. Le motif est composé de trois types de modules ayant trois responsabilités différentes: les modèles, les vues et les contrôleurs.

- Un modèle contient les données à afficher. Cela signifie qu'il est responsable de récupérer les données, de les convertir selon des concepts chargés de sens pour l' application, tels que le traitement, la validation.
- Une vue contient la présentation de l'interface graphique. Elle utilise des données issues du modèle pour fournir une page HTML
- Un contrôleur gère les requêtes des utilisateurs. Elle est responsable de retourner une réponse avec l'aide mutuelle des couches Modèle et Vue.

L'implémentation peut différer un peu selon les cas. Par exemple, dans le cas du Framework Cake PHP[35], le cycle de vie d'une requête HTTP s'organise ainsi:



Une requête utilisateur demande une page ou une ressource dans votre application. Dans cette approche le dispatcher s'occupe de sélectionner le contrôleur concerné.

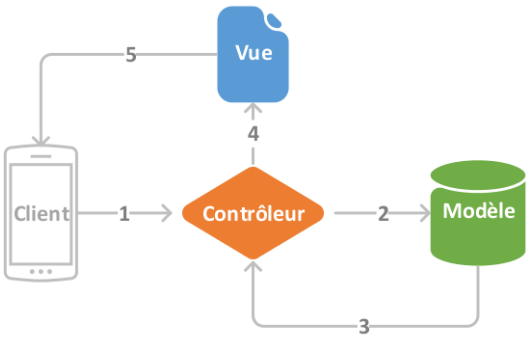
Une fois que la requête arrive au contrôleur, celui-ci va communiquer avec la couche Modèle pour traiter la récupération de données ou les opérations de sauvegarde qui seraient nécessaires.

Cette communication terminée, le contrôleur va donner à un objet vue , la tâche de générer une sortie résultant des données fournies par le modèle.

Finalement, quand cette sortie est générée, elle est immédiatement rendue à l'utilisateur.

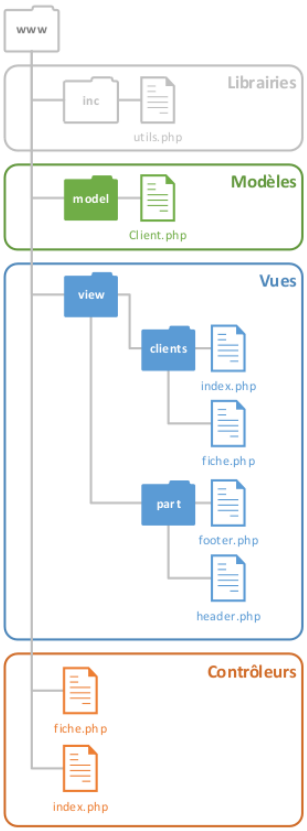
34 https://fr.wikipedia.org/wiki/Mod%C3%A8le-vue-contr%C3%B4leur
35 https://book.cakephp.org/2.0/fr/cakephp-overview/understanding-model-view-controller.html

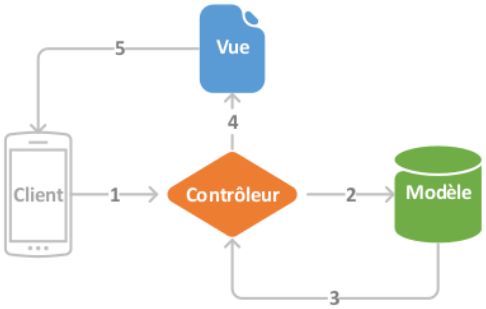
Dans le cadre du cours ICT 133b, le modèle retenu est simplifié[36] et s'organise ainsi:



1. Le client appelle un contrôleur : `http://monsite.com/fiche.php`
2. Le contrôleur demande des données au modèle : `$client = getClient($idClient);`
3. Le modèle transmet les données au contrôleur : `return $requeteSQL->fetch();`
4. Le contrôleur appelle la vue : `require 'view/clients/fiche.php';`
5. La vue construit une page HTML et l'envoie au client : `<h2><?php echo $client['nom'] ?></h2>`

L'arborescence type d'un tel projet devient la suivante :





36 steve.fallet@divtec.ch, septembre 2015

10 Gestion des formulaires

Les formulaires HTML permettent de transmettre des informations du client au serveur. Leur syntaxe et construction à déjà été étudié précédemment.

Il existe deux méthodes pour envoyer des données du navigateur au serveur. L'une envoie les données via la barre d'adresse, l'autre via le corps de la requête. Ces réglages sont réalisés dans la balise HTML form.

- La propriété *action* détermine vers quel script est envoyé le formulaires
- La propriété *method* détermine de quelle manière les données sont envoyées

10.1 Transmission par la méthode GET

Les données transiteront à travers l'URL. Elles seront donc visibles dans la barre d'adresse du navigateur . On peut les récupérer dans le script PHP dans le tableau associatif `$_GET`.

Cette méthode est assez peu utilisée pour la gestion des formulaires car la taille maximale de l'URL est généralement de 2000 caractères. On la retrouve toutefois pour définir des requêtes de recherches ou de filtres notamment pour en permettre la sauvegarde et le partage.

```
<form action="frm1.php" method="get">
    <label for="zNomFamille">Nom:</label>
    <input type="text" name="zNomFamille" id="zNomFamille"><br>
    <label for="zPrenom">Prénom:</label>
    <input type="text" name="zPrenom"><br>
    <button type="submit" name="btnEnvoi" value="btnEnvoi"
id="btnEnvoi">Envoyer</button>
</form>
```

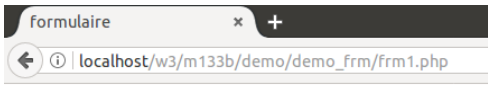
10.2 Transmission par la méthode POST

Les données ne transiteront par le corps de la requête HTTP, l'utilisateur ne les verra donc pas passer dans la barre d'adresse. Elles seront dans le corps de la requête http.

On pourra les récupérer dans le script PHP dans le tableau associatif `$_POST`.

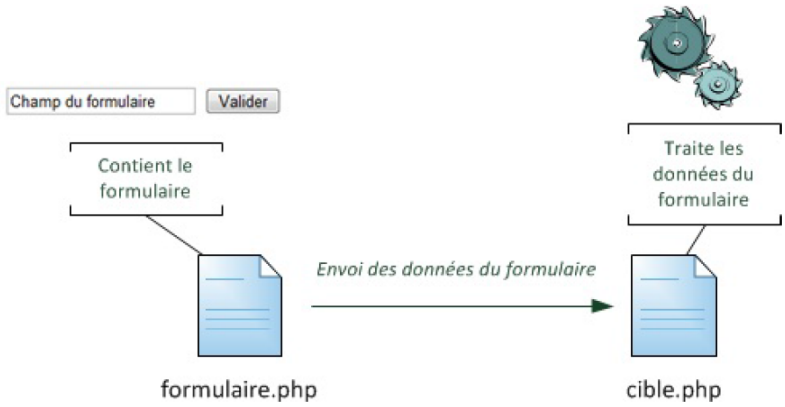
Cette méthode permet d'envoyer autant de données que l'on veut (configuration serveur, défaut de 10MB), ce qui fait qu'on la privilégie le plus souvent. Néanmoins, les données ne sont pas plus sécurisées qu'avec la méthode GET et il faudra toujours vérifier que tous les paramètres sont bien présents. On ne doit pas plus faire confiance aux formulaires qu'aux URL.

```
<form action="frm1.php" method="post">
    <label for="zNomFamille">Nom:</label>
    <input type="text" name="zNomFamille" id="zNomFamille"><br>
    <label for="zPrenom">Prénom:</label>
    <input type="text" name="zPrenom"><br>
    <button type="submit" name="btnEnvoi" value="btnEnvoi"
id="btnEnvoi">Envoyer</button>
</form>
```



10.3 Récupération des données

Principe de base de l'interaction entre le formulaire et le script de traitement du côté serveur.

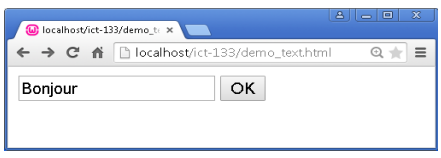


Dans la réalité, il arrive que le formulaire et son traitement soit regroupé dans un même fichier.

La récupération des valeurs transmises par le formulaire est potentiellement dangereuse en terme de sécurité. Il faudra toujours s'assurer que l'internaute ne tente pas de corrompre le fonctionnement du site par ses envois. Il existe plusieurs type de danger. L'injection de code exécutable en est un bon exemple

Exemple – input de type text

```
<form method="get" action="prog.php">
      <p>
            <input type="text" name="v_texte" />
            <input type="submit" value="OK"/>
      </p>
</form>
```

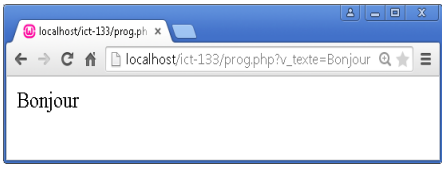


L'envoi du formulaire génère la requête http suivante :

http://localhost/ict-133/prog.php?v_texte=Bonjour

Le script prog.php contient :

```
<?php
      if( $_GET['v_texte']!=''){
            echo $_GET['v_texte'],'<br />';
      }
      else{
            echo 'Rien n\'a été saisi','<br />';
      }
?>
```



Il est habituel de voir du code qui teste l'existence de la variable par

```
if ( isset($_GET['v_texte']) ) …
```

Ceci permet de s'assurer que l'internaute est bien passé par le formulaire pour accéder à la page php.

- La variable $_GET contient toutes les variables passées par la méthode GET
- La variable $_POST contient toutes les variables passées par la méthode POST

ATTENTION : Un champ non-nommé n'est pas passé en paramètre de la requête.

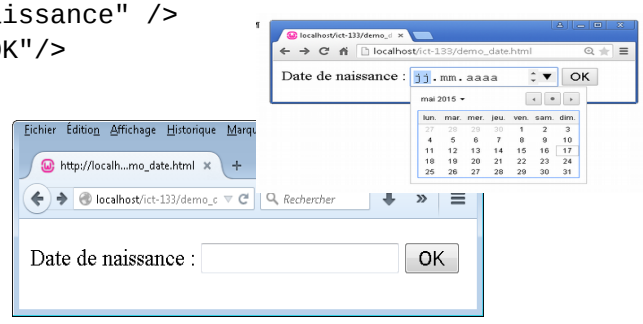
```
<form method="get" action="prog.php">
        <p>
          <input type="text" />
          <input type="submit" value="OK"/>
        </p>
</form>
```

La requête générée sera http://localhost/ict-133/prog.php car aucun champ ne contient de propriété `name`

Exemple – input de type date

ATTENTION : Les navigateurs ne prennent par tous en charge de façon optimale les nouveaux types de champs de saisie. De ce fait, le script php qui recevra les données devra s'assurer de leur bon format. Cette technique s'appelle le contrôle de champ

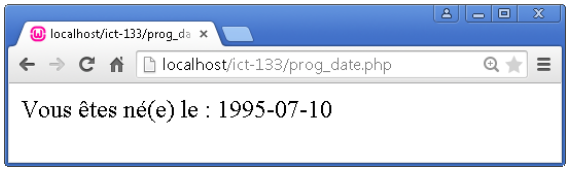
```
<form method="post" action="prog_date.php">
  <p>
    <label>Date de naissance : </label>
    <input type="date" name="v_naissance" />
    <input type="submit" value="OK"/>
  </p>
</form>
```



L'envoi du formulaire génère la requête : http://localhost/ict-133/prog_date.php

```
<?php
  if( $_POST['v_naissance']!=''){
    echo 'Vous êtes né(e) le : ' , $_POST['v_naissance'],'<br />';
  }
  else{
    echo 'Rien n\'a été saisi','<br />';
  }
?>
```

Si l'utilisateur a saisi la date : 10.07.1995, le script affichera :



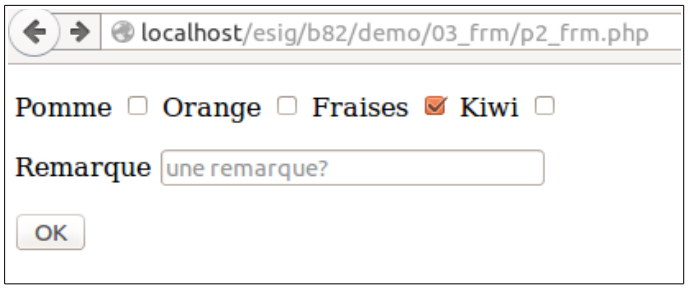
Remarquons ici que les données saisies ne sont pas visibles dans l'url puisque la méthode d'envoi spécifiée est post.

Cas particulier des cases à cocher

Une case à cocher est créé par la balise `<input type="checkbox" .../>`. Pour qu'une case soit cochée par défaut il suffit d'ajouter l'attribut checked="checked" à la case en question.

Un groupe de cases à cocher porte le même nom et est un tableau (ne pas oublier les crochets dans le nom de la variable). A chaque case est associée une valeur. Le script PHP reçoit un tableau qui ne contient que les valeurs des cases cochées.

Attention : Si aucune case n'est cochée, la variable n'existe pas. Le script PHP devrait vérifier l'existence de la variable avant de lire son contenu (isset).



Code du formulaire en html :

```
<form name="frm" method="get" action="p2_reponse.php">
    <p>
    <label for="pomme">Pomme </label>
    <input type="checkbox" id="pomme" name="v_choix[]" value="pomme"/>

    <label for="orange">Orange </label>
    <input type="checkbox" id="orange" name="v_choix[]" value="orange" />

    <label for="fraise">Fraises </label>
    <input type="checkbox" id="fraise" name="v_choix[]" value="fraise"
        checked ="checked"/>

    <label for="kiwi">Kiwi </label>
    <input type="checkbox" id="kiwi" name="v_choix[]" value="kiwi" />
    </p>

    <p>
        <label for="remarque">Remarque </label>
        <input type="text" id="remarque" name="v_remarque" placeholder="une remarque?"
/>
    </p>

    <p><input type="submit"value="OK"/></p>
</form>
```

L'envoi du formulaire génère la requête :

`http://localhost/esig/p2_reponse.php?v_choix[]=pomme&v_choix[]=fraise?v_choix[]=pomme&v_choix[]=fraise`

Script de traitement en php :

```
<?php
      foreach ($_GET['v_choix'] as $cle => $valeur) {
          echo $cle, '--', $valeur, '<br />';
      }

?>
```

```
0--pomme
1--fraise
```

Script PHP avec le contrôle d'existence des variables :

```
<?php

    //contôle préalable de la disponibilité
    if (isset($_GET['v_choix'])) {
        foreach ($_GET['v_choix'] as $cle => $valeur) {
            echo $cle, '--', $valeur, '<br />';
        }
    }
?>
```

```
0--pomme
1--fraise
```

10.4 Contrôle de champs côté serveur

Les données envoyées par un formulaire au serveur servent souvent à peupler ou à interagir avec une bases de données. Il faut par conséquent s'assurer de la cohérences des données reçues. Elle doivent d'une part correspondre au format attendu. Par exemple une date pourrait devoir être au format aaaa-mm-jj et donc ne contenir que de chiffres et des -

D'autre part, les données envoyées doivent respecter des règles et contraintes du métier. Par exemple un mot de passe doit avoir au moins 8 signes dont des lettres et des chiffres.

Les éléments de formulaires des navigateurs actuels offrent un certain nombre de contrôle de champ au niveau du navigateur, notamment pour ce qui concerne la gestion des dates, des adresses de courriel ou des URL. Dans certain cas, comme sur les champs de texte, les navigateurs permettent de définir une validation à l'aide d'empreinte pour assurer le respect d'une règle de contraintes métier

Le contrôle de validation des champs du côté du serveur ne peut toutefois pas être négligé. Il n'est jamais certain que le navigateur utilisé par l'internaute offre les validations du côté client ni que l'envoi se fera par un navigateur. Il peut aussi arriver que se soit une application d'un tout autre ordre que se charge de communiquer avec le serveur.

10.4.1 Quelques technique de validation de champs

Ces techniques usuelles et connues sont essentiellement tirés d'exemple pris sur le internet[37]. Ces solution s'appuient souvent sur les possibilités de PHP ou sur des règles écrites sous la forme d'expression régulière[38] : "une chaîne de caractères que l'on appelle parfois un motif et qui décrit un ensemble de chaînes de caractères possibles selon une syntaxe précise"

Exemple de validation d'un champ obligatoire

```php
//ce champ est obligatoire
if(empty($_POST['v_nom']))
 $erreur['nom']='ATTENTION ce champ est obligatoire';
else
 $nom=$_POST['v_nom'];
```

Exemple de validation d'un champ facultatif

```php
//ce champ est facultatif
if(empty($_POST['v_prenom'])
    $prenom="";
else
    $prenom=$_POST['v_prenom'];
```

37 http://www.w3schools.com/php/php_form_url_email.asp
38 https://fr.wikipedia.org/wiki/Expression_rationnelle

Exemple de validation d'une plage de valeurs

```
$rangeErr="";
$rangeMin=10;
$rangeMax=20;
$value=21;
if ($value<$rangeMin || $value>$rangeMax) {
    $erreur['plage'] = "hors plage";
}
```

Exemple de validation d'une chaîne de caractère

```
$nameErr="";
$name="Dupond";
if (!preg_match("/^[a-zA-Z ]*$/",$name)) {
    $erreur['nom'] = "Only letters and white space allowed";
}
```

Exemple de validation d'une adresse de courriel

```
$emailErr="";
$email = 'a@a.ch';
$email = 'a@a@.ch';
if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
    $erreur['courriel'] = "Invalid email format";
}
```

Exemple de validation d'une date

```
$dateErr="";
$date="2009-1212";
$date="2009-12-12";
if ( !preg_match("/^[0-9]{4}-(0[1-9]|1[0-2])-(0[1-9]|[1-2][0-9]|3[0-1])$/",$date)
){
    $erreur['date']= "La date n'est pas valide";
}
```

Exemple de validation d'un n° de téléphone Suisse

```
$telErr="";
$tel="032 422 43 45";
$tel="++41 32 422 43 45";
if ( !preg_match("/^([0-9]{3}|[+]{1,2}41(\s|-)[1-9]{2})(\s|-)[0-9]{3}(\s|-)[0-9]
{2}(\s|-)[0-9]{2}$/",$tel) )
{
    $erreur['téléphone']= "Le numéro de téléphone n'est pas valide";
}
```

Exemple de traitement du résultat dans une liste :

```
foreach($erreur as $key => $value){
  echo'<li>', $value,'</li>';
}
```

Tous ces exemples sont à essayer dans le fichier de démo associé. La plupart peuvent se retrouver sur Internet.

10.5 Formulaire sur une page

Pour faciliter la gestion du contrôle de champs du côté serveur, il est parfois utile de rassembler le formulaire et la logique de contrôle dans le même fichier. Le fonctionnement

du formulaire diffère alors un peu

Le formulaire contient une partie de traitement qui ne s'exécute que si le formulaire à été envoyé. Pour le découvrir, il suffit de tester la présence de valeurs dans la variable $_POST ou dans la variable $_GET. Il est courant de créer une variable cachée dans le formulaire de manière à savoir avec certitude qu'elle existera lors de l'envoi du formulaire.

```
<body>

<?php
    //seulement si on est passé par là
    if(isset($_POST['v_frm_identite']))
    {
        if(empty($_POST['v_prenom']))
            $prenom="";
        else
            $prenom=$_POST['v_prenom'];

        if(empty($_POST['v_nom']))
            $nom="";
        else
            $nom=$_POST['v_nom'];
    }
?>
    <form action="<?php echo $_SERVER['PHP_SELF']  ?>" method="post">
    <fieldset>
        <p>
            <label>Prénom : </label>
            <input type="text" name="v_prenom" value="" />
        </p>
        <p>
            <label>Nom : </label>
            <input type="text" name="v_nom" value="" />
        </p>
        <p>
            <button type="submit">Envoyer</button>
            <input type="hidden" name="v_frm_identite" value="frm_identite"
        </p>
    </form>
    </fieldset>
    <?php
        //affichage si le formulaire à été envoyé
        if(isset($_POST['v_frm_identite']))
        {
            echo '<p>','Prénom: ',$prenom,'</p>';
            echo '<p>','le nom est :',$nom,'</p>';
        }
    ?>
</body>
</html>
```

Zone de traitement des données.
On y passe que si le formulaire à déjà été envoyé au moins une fois.
Ne s'exécute pas au premier chargement de la page

Variable cachée passée au serveur lors de l'envoi du formulaire

Zone d'affichage.
On y passe que si le formulaire à déjà été envoyé au moins une fois.
Ne s'exécute pas au premier chargement de la page

10.6 Rémanence

Lorsque l'internaute saisit les données d'un formulaire puis le soumet, il se peut qu'il ait oublié de saisir des données obligatoires ou que certaines données ne respectent pas les contraintes définies.

Dans ce cas, le programme PHP doit afficher des messages adéquats afin que l'utilisateur complète et corrige ses saisies.

La rémanence consiste à ré-afficher dans le formulaire les données valides déjà saisies par l'internaute.

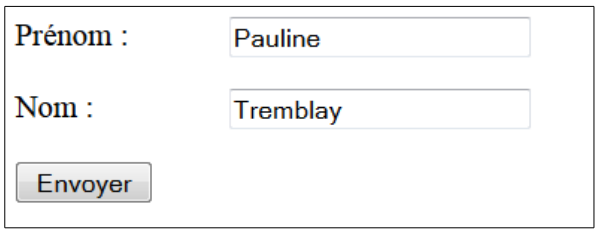
10.6.1 Principe général

Le formulaire est intégré dans le script PHP. L'attribut action de la balise <form> fait appel au script lui-même. On peut indiquer le nom du script en dur dans l'attribut action. On peut aussi utiliser pour cela la clé PHP_SELF de la variable superglobale $_SERVER[39].

```
<form action="<?php echo $_SERVER['PHP_SELF'] ?>" method="post">
```

Lors de l'envoi du formulaire, le script PHP contrôle si toutes les données attendues sont présentes et cohérentes. Si c'est le cas, le traitement prévu est effectué.

```
if (!empty($_POST['v_nom']) && !empty($_POST['v_prenom'])) {
    echo '<p>Bonjour ', $_POST['v_prenom'], ' ', $_POST['v_nom'], '</p>';
}
```



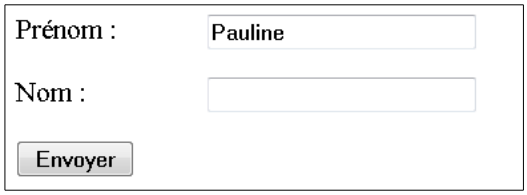
Résultat après clic sur le bouton Envoyer

Bonjour Pauline Tremblay

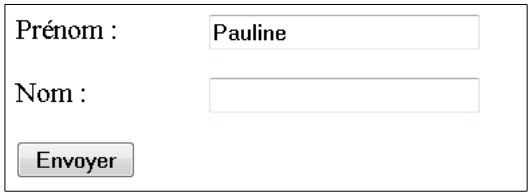
Si ce n'est pas le cas, on ré affiche ce qui a déjà été saisi dans le formulaire en agissant sur l'attribut *value*

39 Pour des aspects de sécurité, cf. http://www.w3schools.com/php/php_form_validation.asp

```
<p>
    <label>Prénom * : </label>
    <input type="text" name="v_prenom"
           value="<?php
                        if (!empty($_POST['v_prenom']))
                            echo $_POST['v_prenom'];
                  ?>"
    />
</p>
```



Résultat après clic sur le bouton Envoyer



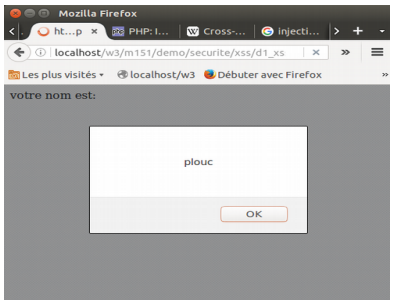
Code complet de cet exemple :

```
<body>
<?php
if (!empty($_POST['v_nom']) && !empty($_POST['v_prenom'])) {
    echo '<p>Bonjour ', $_POST['v_prenom'], ' ', $_POST['v_nom'], '</p>';
}
else
{
?>
    <form action="<?php echo htmlspecialchars($_SERVER['PHP_SELF']) ?>" method="post">
        <p>
            <label>Prénom * : </label>
            <input type="text" name="v_prenom"
                   value="<?php
                                if (!empty($_POST['v_prenom']))
                                    echo $_POST['v_prenom'];
                           ?>"
            />
        </p>
        <p>
            <label>Nom *: </label>
            <input type="text" name="v_nom"
                   value="<?php
                                if (!empty($_POST['v_nom']))
                                    echo $_POST['v_nom'];
                           ?>"
            />
        </p>
        <p>
            <button type="submit">Envoyer</button>
        </p>
    </form>
<?php
}
?>
</body>
```

11 Injection XSS

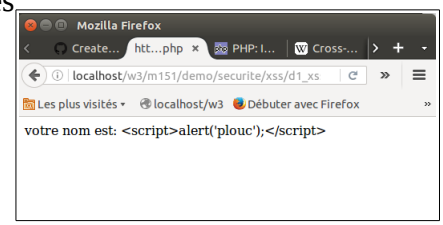
Les injections XSS[40] consistent à insérer une ligne de script JavaScript dans un champ de formulaire ou dans L'URL de la page. Lorsque le script PHP retourne ce contenu dans le flux HTML, le navigateur l'interprète comme du code exécutable du côté client.

```
<?php
//$nom="Perret";
$nom="<script>alert('plouc');</script>"
?>
<p>votre nom est: <?php echo $nom;?></p>
```

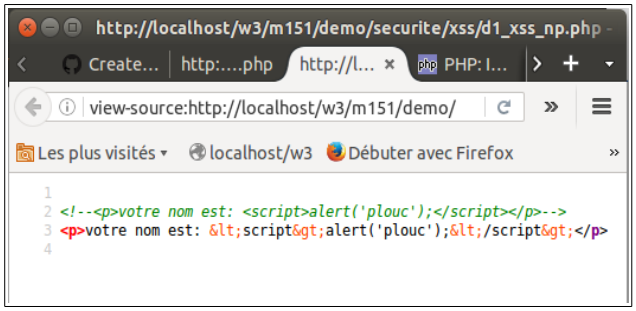


La solution pour se protéger consiste à transformer les caractères spéciaux de la chaîne à afficher en entités HTML :

```
<?php
//$nom="Perret";
$nom="<script>alert('plouc');</script>"
?>
<p>votre nom est: <?php echo htmlentities($nom);?></p>
```



La source de la page permet de visualiser les entités en question.



Lorsque les données potentiellement infectées du formulaires sont stockées dans une base de données ou un fichier, l'injection devient permanente et peut mettre l'entier de l'application web à genoux.

Pour s'en protéger, les données issues du formulaire sont stockées telles quelles dans la base de données ou le fichier. Au moment de les injecter dans le flux HTML, il faut impérativement appliquer la séquence d'échappement des données en les transformant en

40 https://fr.wikipedia.org/wiki/Cross-site_scripting

entités à l'aide des fonctions `htmlentities()`[41], `htmlspecialchars()`[42] ou encore à l'aide des fonction de filtres[43] tels que `filter_var()` ou `filter_input()` associées au filtres de nettoyages[44] `FILTER_SANITIZE_SPECIAL_CHARS` ou `FILTER_SANITIZE_STRING`

Une autre approche consiste à simplement supprimer tous les tags présents dans la chaîne. C'est la fonction `strip_tag`[45] qui s'en charge. Cette fonction est paramétrable. Elle permet de définir quels tag sont malgré tout acceptables.

Il existe sur Internet différents outils de vérification et de visualisation des effets de l'injection XSS. Le site de Steve Fallet[46] montre notamment l'effet des différentes fonction de protection proposées par PHP.

41 http://php.net/manual/fr/function.htmlentities.php
42 http://php.net/manual/fr/function.htmlspecialchars.php
43 http://php.net/manual/fr/ref.filter.php
44 http://php.net/manual/fr/filter.filters.sanitize.php
45 http://php.net/manual/en/function.strip-tags.php
46 https://kode.ch/tests/injections.php

12 Téléversement de fichier

La possibilité d'envoyer une image ou un fichier au serveur s'appuie sur la fonction d'envoie de données entre le client et le serveur Web. Pour réussir cela il faut tout d'abord vérifier que le interpréteur de commande sur le serveur accepte de le faire. Le fichier `php.ini` du serveur contient les informations suivantes:

php.ini

```
;;;;;;;;;;;;;;;;;;
; File Uploads ;
;;;;;;;;;;;;;;;;;;

; Whether to allow HTTP file uploads.
; http://php.net/file-uploads
file_uploads = On

; Temporary directory for HTTP uploaded files (will use system default if not
; specified).
; http://php.net/upload-tmp-dir
;upload_tmp_dir =

; Maximum allowed size for uploaded files.
; http://php.net/upload-max-filesize
upload_max_filesize = 2M

; Maximum number of files that can be uploaded via a single request
max_file_uploads = 20
```

La taille par défaut des fichiers envoyés ne peut excéder 2Mo.

12.1 Formulaire d'envoi

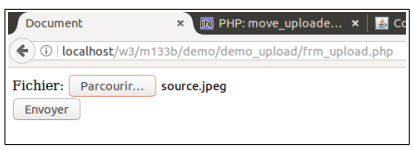
Le fichier est sélectionné sur le client à l'aide d'une formulaire et d'un élément de formulaire capable de parcourir les disques du client et de sélectionner un fichier.

```
<form action="do_upload.php" method="post" name="form1"  enctype="multipart/form-data">
    <div>
        <label for="zch">Fichier:</label>
        <input name="image" type="file" id="zch"/>
    </div>
    <div>
        <button type="submit" name="Envoyer" value="upload">Envoyer</button>
    </div>
</form>
```

Il faut constater ici que la balise `form` contient une prpriété `enctype="multipart/form-data"`. Cette propriété est primordiale à l'envoie de données via un formulaire[47].

Dans ce cas, l'élément de formulaire permettant le choix du fichier est de type FILE et se nomme `image`. C'est grâce à cet identifiant qu'il sera possible de manipuler les données sur le serveur.

Ce qui donne un formulaire comme celui-ci:



47 cf. http://www.w3schools.com/tags/att_form_enctype.asp

12.2 Récupération des données sur le serveur

Le principe de récupération des données dans PHP est assez simple. Lors de l'envoi du formulaire depuis le client Web, les données ont transitées vers le serveur web et le fichier à été stocké dans un répertoire temporaire du serveur sous un nom particulier. Le script de traitement du formulaire devra dans un premier temps déplacer le fichier envoyé vers une destination choisie sur le serveurs. Il s'agit souvent dans la pratique d'un sous répertoire *./fichiers* ou *./images*.

La variable super globale de traitement des chargements vers le serveur s'appelle £_FILE. C'est une superglobale qui contiendra un tableau associatif relié à chaque élément de type file du formulaire html.

Dans le $_FILE:

```
array (size=1)
  'image' =>
    array (size=5)
      'name' => string 'source.jpeg' (length=11)
      'type' => string 'image/jpeg' (length=10)
      'tmp_name' => string '/tmp/phprxBBpZ' (length=14)
      'error' => int 0
      'size' => int 6069
```

Le premier tableau contient un tableau nommé *image* comme l'élément de formulaire permettant le choix du fichier sur le client. Cet élément *image* est un tableau qui contient les statuts du téléchargement. On reconnaît cinq champ dont le nom temporaire du fichier sur le serveur et le nom réel du fichier.

12.3 Déplacement du fichier vers son répertoire définitif

Il faudra utiliser une fonction spécifique pour déplacer le fichier de son emplacement temporaire vers son emplacement définitif:

```
$uneImage = '';
$res = move_uploaded_file($_FILES['image']['tmp_name'],'./img/'.$_FILES['image']
['name']);
if ($res === true) {
    $uneImage = $_FILES['image']['name'];
}
```

L'affichage de l'image dans le corps de la page se fait de la manière suivante :

```
<img src="<?php echo './img/'.$uneImage; ?>">
```

Fonction de téléchargement sécurisée :

L'envoi de fichier sur le serveur reste une action potentiellement dangereuse. Il s'agit de rester vigilant à la qualité du fichier téléversé. Commet s'assurer que le fichier ne contient pas du code malicieux ? Comment s'assurer que le fichier n'en cache pas un autre ? Pour se protéger de ces menaces, il faudra améliorer toujours adapter le code de du chargement[48].

48 https://openclassrooms.com/courses/protegez-vous-efficacement-contre-les-failles-web/la-faille-upload-1
http://stackoverflow.com/questions/256172/what-is-the-most-secure-method-for-uploading-a-file

13 Sauvegarde des données de l'utilisateur

13.1 Sur le client avec des cookies

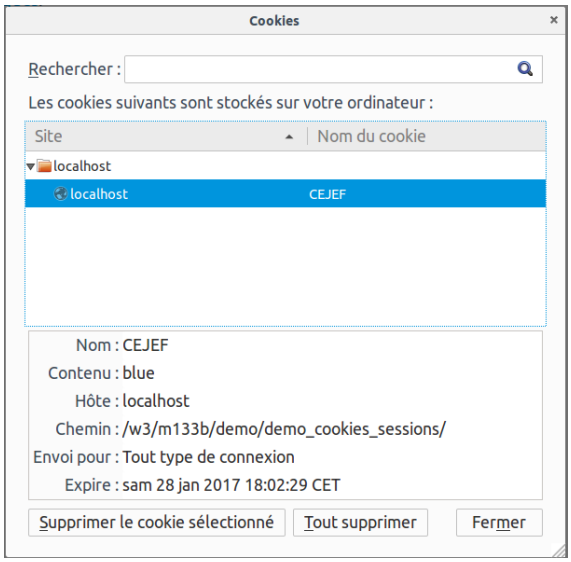
Le protocole http, dans sa version 1.1 reste un protocole sans état (stateless), cela signifie ne garde aucune mémoire des requêtes d'un internaute. Chaque requête, quelle qu'elle soit est vue comme nouvelle et indépendante.

Pour développer des sites plus conviviaux il est nécessaire de garder un certain nombre d'informations liées à un utilisateur (ses préférences, le contenu de son panier d'achat, etc) sur le serveur durant un temps limité configuré dans le fichier php.ini.

Le cookie est un fichier stocké sur le disque de l'utilisateur. C'est le serveur qui demande la création du cookie. Un cookie contient un ensemble d'informations choisies et définies par un serveur. Les données sont liées à un domaine. La rfc2965 définit quelques limites aux cookies dans son chapitre 5.3[49]

- Limitations minimales attendues pour le navigateurs
- Le nombre total minimum de cookie par navigateur est de 300 ;
- La taille maximale d'un cookie est de 4 ko ;
- Il exister au minimum 20 cookies par domaine.

La manière de stocker les cookie sur le client dépend du navigateur utilisé. Dans le cas de Firefox, ils sont stocker dans une base de donnée SQLite éditable via les préférences :



Création de cookies en PHP

La création se fait avec la procédure setcookie :

```
<?php
```

49 cf. http://www.ietf.org/rfc/rfc2965.txt

```
    setcookie('CEJEF', 'blue',(time() + (3600 * 24)) );
?>
```

Ce cookie nommé CEJEF aura pour valeur *blue* et aura une durée de vie de 24 heures.

Règle importante : l'appel de la fonction setcookie doit se faire avant tout envoi au navigateur (instruction echo, balise html...) sous peine de générer une erreur de type *Headers already sent by...*. Cette erreur signifie que la page a déjà été envoyée au navigateur avant de vouloir envoyer des entêtes HTTP.

Récupération des données

La récupération de la valeur d'un cookie se fait à l'aide de la superglobale $_COOKIE[…]

```
$var = $_COOKIE['CEJEF'];
```

$var contiendra la valeur associé avec le cookie CEJEF, donc **blue**

Modification des données

On utilise l'instruction setcookie en fournissant une nouvelle valeur :

```
setcookie('CEJEF', 'red',time()+3600*24);
```

Suppression

On utilise setcookie en ne fournissant le nom et une durée de 1 seconde :

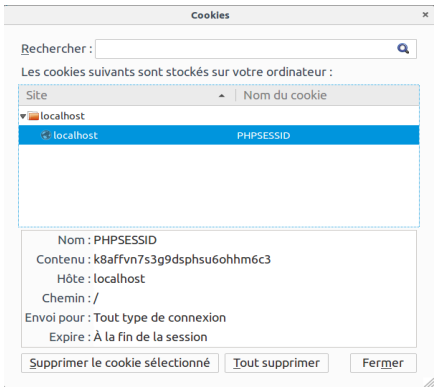
```
setcookie('CEJEF', '', 1);
```

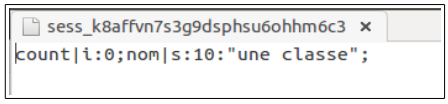
13.2 Sur le serveur avec des sessions

Les sessions[50] sont un moyen simple de stocker des données individuelles pour chaque utilisateur en utilisant un identifiant de session unique. Elles peuvent être utilisées pour faire persister (garder en mémoire) des informations entre plusieurs pages. Les identifiants de session sont normalement envoyés au navigateur via des cookies de session, et l'identifiant est utilisé pour récupérer les données existantes de la session. L'absence d'un identifiant ou d'un cookie de session indique à PHP de créer une nouvelle session, et génère ainsi un nouvel identifiant de session.

La session est souvent stockée sous forme de fichier sur le serveur. Le nom de ce fichier est composé du numéro inscrit dans le cookie du coté serveur :

50 cf. http://php.net/manual/fr/book.session.php





Les sessions suivent une cinématique simple. Lorsqu'une session est démarrée, PHP va soit récupérer une session existante en utilisant l'identifiant de session passé (habituellement depuis un cookie de session) ou si aucun identifiant de session n'est passé, il va créer une nouvelle session.

PHP va ainsi peupler la variable superglobale `$_SESSION` avec toutes les données de session une fois la session démarrée. Lorsque PHP s'arrête, il va prendre automatiquement le contenu de la variable superglobale `$_SESSION`, le linéariser, et l'envoyer pour stockage au gestionnaire de sauvegarde de session.

Les sessions peuvent être démarrées manuellement en utilisant la fonction `session_start()`. Les sessions s'arrêtent automatiquement lorsque PHP a terminé d'exécuter un script, mais peuvent être stoppées manuellement en utilisant la fonction `session_write_close()`.

Écriture dans la variable de session

```php
session_start();

if (isset($_SESSION['count']) === false) {
    $_SESSION['count'] = 0;
    //header("'refresh:5; url=p41_setsession.php");
} else {
    $_SESSION['count']++;
}
```

Tout type de variable peut être mis dans une session, y compris un tableau. C’est commode si on veut mémoriser le contenu d’un panier d’achat.

Règle importante : l'appel de la fonction `session_start()` doit se faire avant tout envoi au navigateur (instruction echo, balise html...) sous peine de générer une erreur de type `Headers already sent by....` Cette erreur signifie que la page a déjà été envoyée au navigateur avant de vouloir envoyer des entêtes HTTP.

Effacement et suppression de la variable de session

L'effacement se réalise en écrivant une valeur vide dans la clé désirée. La suppression avec

`unset` permet d'enlever une clé du gestionnaire de session.

```php
<?php
if (isset($_SESSION['count']) === true) {
    // Vide la valeur de la clé.
    $_SESSION['count'] = '';

    // Efface une clé de la session.
    unset($_SESSION['count']);

    // Détruit toutes les variables de la session courante.
    session_unset($_SESSION['count']);

    // Suprimme la session elle-même.
    session_destroy();
}
?>
```

14 Références et ressources

Ouvrages et revues

[ICT133] Collectif, "module ICT 133", CPLN, Neuchâtel, 2015

[ED-06] Eric Daspet, "PHP5 avancé" (3e édition), Eyrolle, 2006, Paris

Ressources en ligne

[PHP] Manuel de PHP [en ligne]
http://php.net/manual/fr/langref.php
http://www.php.net/manual/fr/mysqli.summary.php
http://php.net/manual/fr/language.exceptions.php

 Manuel de PHP [en ligne]
la plupart de liens sur le manuel sont directement insérer en bas de pages

[SFA15] Standard s d'écriture applicable dans le contexte de l'EMT [en ligne]
https://github.com/fallinov/151A/wiki

[PHP-07] tutoriel PHP[en ligne]
http://www.apprendre-php.com/tutoriels/tutoriel-15-faire-une-redirection-vers-une-autre-page.html