PHP

A. Atlas

C’est Quoi PHP

- L’un des langages de programmation les plus utilisé pour la programmation web
- Créer en 1994 par Rasmus Lerdorf
- Open source
- Permet de creer des pages web dynamiques
- PHP est exécuté (interpreté) coté serveur
- Utilisé dans plusieur système de gestion de contenu CMS(content management systems) comme Wordpress, Drupal and Joomla
- Plusieurs Freamwork robustes écrit en PHP : Laravel, Symfony ...

Installation

Pour réaliser les différentes section de ce cours on à besoin de :

1. Un serveur web : apache
2. Un interpréteur PHP
3. Un serveur de base de donnée mySql

Les trois softwares sont installable à l'aide d'un seul package: XAMP

Cross-platform (X), Apache web server (A), MarieDB database server (M), PHP (P) and Perl (P)

php.ini

- Au cours de développement on a besoin de voire les erreurs dans notre navigateur. Pour cela il faut apporter quelques modification à ***/opt/lampp/etc/php.ini***
- php.ini (fichier de configuration de php)
- Ajouter ce deux lignes à la fin du fichier
  - error_reporting=E_ALL
  - display_errors=On
- Enregistrer et redémarrer apache

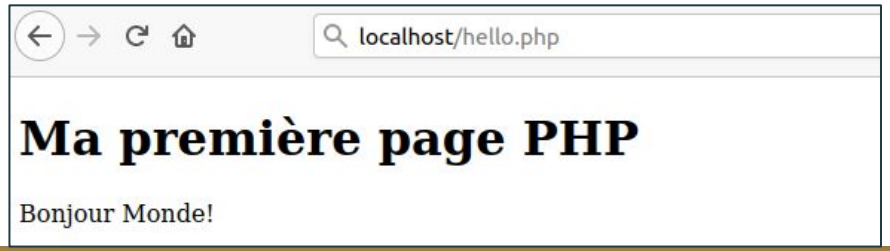
Page Zero

- Verifiez que apache est ON
- Dans le dossier ***htdocs*** (/opt/lampp/htdocs) créez un fichier ***hello.php*** avec le contenu ci-contre.
- Dans votre navigateur visitez le lien ***http://localhost/hello.php***

/opt/lampp/htdocs/tps/hello.php

```
<!DOCTYPE html>
<html>
    <head>
        <title>My first PHP page</title>
    </head>
    <body>
        <h1>Ma première page PHP</h1>
        <?php
            # un commentaire php
            echo "Bonjour Monde!";
        ?>
    </body>
</html>
```

Code interpreter par PHP (***<?php .... ?>***) et transformer en HTML

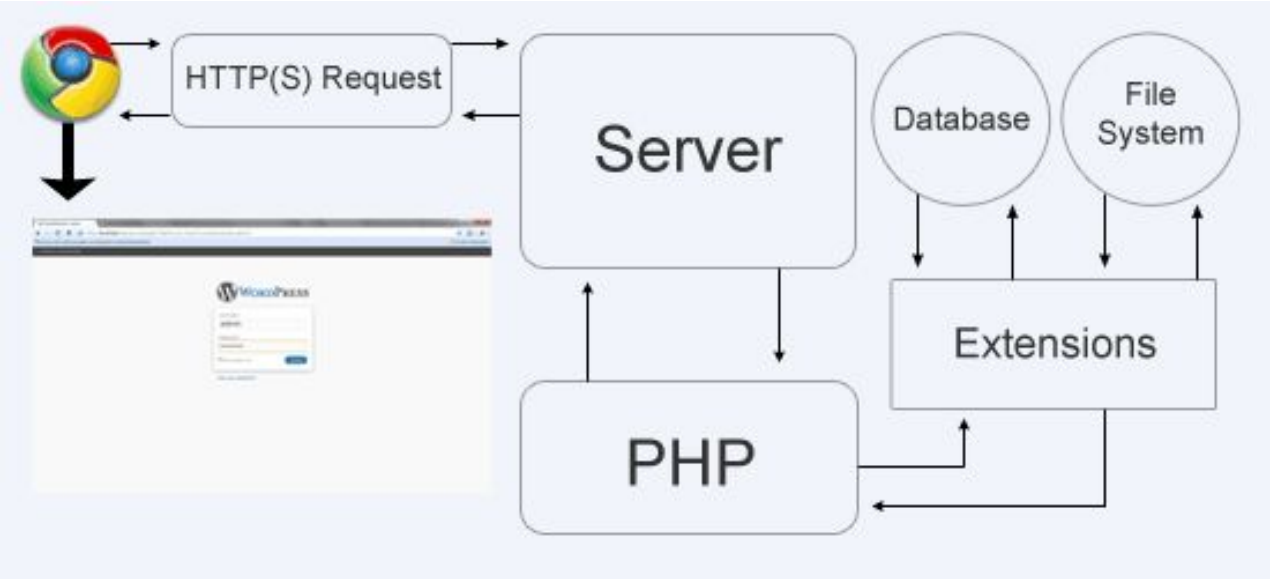


Page Zero

Fichier reçu par le navigateur

```
1 <!DOCTYPE html>
2 <html>
3     <head>
4         <title>My first PHP page</title>
5     </head>
6     <body>
7         <h1>Ma première page PHP</h1>
8         Bonjour Monde!    </body>
9 </html>
```

PHP workflow



Source: https://padhmanaban.com/workflow-of-php/

Php: Output

/opt/lampp/htdocs/outputs.php

```php
<?php
    echo '<h3>Bonjour tous !!!</h3>';
    echo ('un text <br>');
    echo "bonjour ", "tous", "<br>";
    print "Une autre output avec 'print'<br>";
    print " \\ est utilisé pour echapper des caractère spéciaux comme \\ et \" ";
```

Réutilisation du code

- Dans PHP les script sont généralement indépendants les uns des autres.
- Pour pouvoir inclure un code HTML ou PHP d'un fichier dans un autre fichier on utilise:
  - ***include*** : inclure un fichier avec un ***warning*** si ce dernier est ***introuvable***
  - ***require***: inclure un fichier avec une ***Erreure*** si ce dernier est ***introuvable***
  - ***Include_once:*** idem que ***include*** mais évite l'inclusion d'un fichier plusieurs fois
  - ***Require_once:*** idem que ***require*** mais évite l'inclusion d'un fichier plusieurs fois

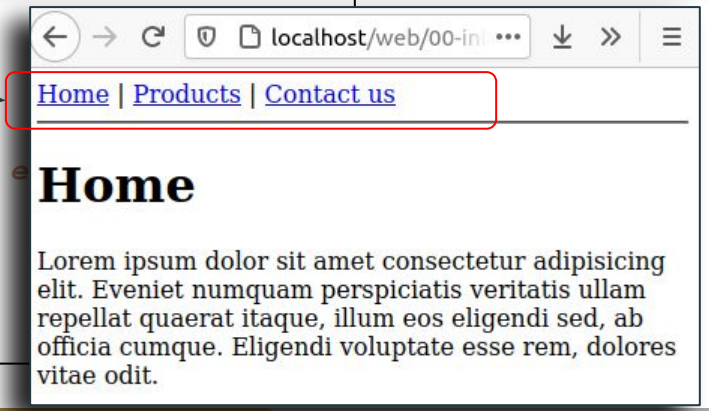
Include

htdocs/home.php

```php
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Home</title>
</head>
<body>
    <?php
    include "./views/header.html";
    echo "<h1> Home </h1>";
    echo "Lorem ipsum dolor sit amet consectetur adipisicing e
    ?>
</body>
</html>
```

htdocs/views/header.html

```html
<a href="home.html">Home</a> |
<a href="products.htmal">Products</a> |
<a href="contact.html">Contact us</a>
<hr>
```



Redirection des utilisateurs: La fonction header()

- L'appel de la fonction ***header()*** doit être effectué ***avant*** tout autre instruction qui génère une sortie au navigateur:

```
<?php

header('Location: http://google.com');

?>

<!DOCTYPE html>

….
```

Le chargement de ce document redirrigera l'utilisateur vers http://google.com

Redirection des utilisateurs: La fonction header()

Utilisation ***inapproprié*** de la fonction ***header()***

Exemple 1

```
<?php
echo 'Hello';
header('Location: http://example.com');
?>
```

Exemple 2

```
<p>This is some text.</p>
<?php
header('Location: http://example.com');
?>
```

Ces exemples peuvent passer sans problème en développement local mais pas dans un serveur de production

PHP: Constantes

- En PHP en peut définir une constante à l'aide de la fonction ***define(nom, valeur)***

```
define("MY_CONST", 1234);
echo MY_CONST;
echo '<BR>';
define("MY_CONST", 5678); # warning: Constant MY_CONST already defined ...
echo MY_CONST;  # 1234
```

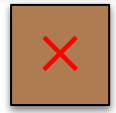
- La valeur de ***MY_CONST*** reste inchangé dans tout le scripte

PHP: Variables

- Les variables servent à stocker des données
- Le contenue d'une variable est modifiable
- En PHP le nom d'une variable commence par **$** suivi d'une alphabet puis tout autre caractère alphanumérique et _ (A-z, 0-9, and _)
- Les noms des variables sont sensibles à la casse $nom et $Nom sont deux variables

$nom
$elem1
$my_name
$e1245

nom
$1elem
$my var
$my_name!



Essayez toujours d'utiliser des noms de variables significatifs et évitez des nom comme $a, $b ..

PHP: Variables

```
<?php
        $taille= 76; # déclaration et initialisation d'une variable
        echo $taille; # afficher le contenue d'une variable
         $taille= 176; # modifier le contenu de la variable $taille
        echo 'Ma taille est: $taille'; # affiche: Ma taille est $taille !!!
        echo "Ma taille est: $taille"; # affiche: Ma taille est 176
?>
```

Data types

- En PHP une variable peut être utiliser pour stocker différent type de valeures:
  - ***int*** ou ***integer***
  - ***float*** ou ***double***
  - ***bool*** ou ***boolean***
- Pour verifier le type d'une variable on utiliser la fonction ***var_dump($nomVar)***
- Pour changer le type d'une variable (casting) on ajoute (int); (float) ou (bool) avant le nom de la variable
  - ***$a= 2.7***
  - ***var_dump($a)*** *# affiche float(2.7)*
  - ***(int)$a***
  - ***var_dump($a)*** *# affiche int(2)*
- On peut aussi utiliser (string); (array) ou (object)

Les opérateurs de comparaison

Operator	Name	Example	Explanation
==	Equal	$a == $b	If $a is equal to $b, returns true
===	Identical	$a === $b	If $a is equal to $b and the same type, returns true
!=	Not equal	$a != $b	If $a is not equal to $b ,returns true
<>	Not equal	$a <> $b	If $a is not equal to $b ,returns true
!==	Not identical	$a !== $b	If $a is not equal to $b and not the same type, returns true
>	Greater than	$a > $b	If $a is greater than $b, returns true
<	Less than	$a < $b	If $a is less than $b, returns true
>=	Greater than or equal to	$a >= $b	If $a is greater than or equal to $b , returns true
<=	Less than or equal to	$a <= $b	If $a is less than or equal to $b, returns true
< = >	Spaceship	$a <=> $b	returns an integer less than, equal to, or greater than zero, depending on if $a is less than, equal to, or greater than $b.

Les opérateurs logique

Operator	Name	Example	Explanation
and	And	$a and $b	Return true if $a and $b both are true
or	Or	$a or $b	Return true if either $a or $b is true
xor	Xor	$a xor $b	Return true if either $ or $b is true but not both
&&	And	$a && $b	Return true if $a and $b both are true
||	Or	$a || $b	Return true if either $a or $b is true
!	Not	!$a	Return true if $a is not true

Les opérateurs de chaînes de caractères

Operator	Name	Example	Explanation
.	Concatenation	$a . $b	Concatenation of $text1 and $text2
.=	Concatenation and assignment	$.= $b	appends $text2 to the $text1

Les opérateurs d'affectation

Operator	Like as..	Explanation
A = B	A = B	Assign value from right side operand to left side.
A += B	A = A + B	Add both operands and then assign to left side
A -= B	A = A - B	Subtract the operands and result assign to left side
A *= B	A = A * B	Multiply both operands and then result assign to left side
A /= B	A = A / B	Division of operands and result assign to left side
A %= B	A = A % B	Modulus of two operands and assign to left side

Les opérateurs conditionnels

Operator	Name	Example	Explanation
?:	Ternary	$a = *expr1* ? *expr2* : *expr3*	Return the value of $a. The value of $a is *expr2* if *expr1* = TRUE. The value of $a is *expr3* if *expr1* = FALSE
??	Null coalescing	$a = *expr1* ?? *expr2*	Returns the value of $a. The value of $a is *expr1* if *expr1* exists, and is not NULL. If *expr1* does not exist, or is NULL, the value of $a is *expr2*.

Les opérateurs d'i

Operator	Name	Explanation
++$a	Pre-Increment	Increase the value of $a by one, returns $a
$a++	Post-increment	Returns $a, then increase value of $a by one
--$a	Pre-decrement	Decrease the value of $a by one, returns $a
$a--	Post-decrement	Returns $a, then decrease value of $a by one

https://demonuts.com/php-operators/

Les chaînes de caractères

```
$msg = 'Hello';

$greeting = "Good Morning";

$emptyStr = "";

$areacode = "(208)";

$contact = '+1' . $areacode . '1234567';

echo $contact;
```

Les chaînes de caractères

```
$str1 = 'Good Day!';

echo strlen($str1);

$str2 = 'Hello World';

$str3 = strtolower($str2);

$str4 = strtoupper($str2);

echo '<BR>'.$str2;

echo '<BR>'.$str3;

echo '<BR>'.$str4;
```

```
$str5 = ' is ';

echo 'PHP'.$str5.'Fun<BR>';

echo 'PHP'.trim($str5).'Fun<BR>';

$str6 = '**Hello**World***';

echo trim($str6, '*');

$str7 = 'ABCDEF';

echo substr($str7, 2).'<BR>';

echo substr($str7, -3).'<BR>';

echo substr($str7, 2, 1);
```

Les chaînes de caractères

```
echo strtotime("next Monday");

echo date('d-M-Y', strtotime("+ 10 hours"));

echo date('d-M-Y', strtotime("+ 10 hours"));
```

Dans php.ini : date.timezone=America/New_York

```
date_default_timezone_set('Africa/Casablanca');
```

Arrays

Tableaux indexés

```
$firstArr = array();

$secondArr = array(11, 16, 4, 9, 12);

$fruitsArr = array('Apple', 'Banana', 'Coconut');

echo $secondArr[3]; # Accéder au quatrième ’élément

$secondArr[3] = 20; # Modifier le quatrième ’élément
```

Tableaux associatifs

```
$assocArr = array(

‘key1’ => 11,

'key2' => 16,

'key3' => 12

);

echo $assocArr['key2'];
```

Arrays

```
$simpleMDArr = array(

array(1, 2, 1, 4, 5),

array(0, 5, 1, 3, 4),

array(4, 1, 7, 8, 9)

);

echo $simpleMDArr[2];

echo $simpleMDArr[2][3];
```

```
$assocMDArr = array(

"first array" => array(1, 2, 6, 1, 3),

"second array" => array(3, 5, 1, 8, 9),

"third array" => array(1, 0, 9, 4, 7)

);

echo $assocMDArr["first array"];

echo $assocMDArr["first array"][2];
```

Array

```
$anotherAssocMDArr = array(
"first player" => array("name" => 'John', "age"
=> 25),
"second player" => array("name" => 'Tim', "age"
=> 35)
);
echo $anotherAssocMDArr["first player"];
echo $anotherAssocMDArr["first player"]["age"];
```

Arrays

```
$myArray = array(2, 5.1, 'PHP', 105);

var_dump($myArray); # array(4) { [0]=> int(2) [1]=> float(5.1) [2]=> string(3) "PHP" [3]=> int(105) }

print_r($myArray); # Array ( [0] => 2 [1] => 5.1 [2] => PHP [3] => 105 )
```

Arrays

- Ajouter un élément

```
$addDemo = array(1, 5, 3, 9);

$addDemo[] = 7;

$addDemoAssoc = array('Peter'=>20, 'Jane'=>15);

$addDemoAssoc['James'] = 17;
```

- Supprimer un élément

```
$colors =

array("red", "black", "pink", "white");

array_splice($colors, 2);
```

```
$awardwinners = array(

"Gold"=>"Max",

"Silver"=>"Boots",

"Bronze"=>"Dora");

array_splice($awardwinners, 1);

$pets = array("corgi", "poodle", "golden retriever","jack russell");

array_splice($pets, 1, 2);
```

Les structures de contrôle en PHP

```
if( condition1 ){
        //instructions ...
}elseif( condition2 ){
        //instructions ...
}elseif( condition3 ){
        //instructions ...
}else{
        //instructions ...
}
```

```
switch( variable){
        case  val1:  // instructions…; break;
        case  val2:  // instructions…; break;
        case  val3:  // instructions…; break;
        ..
        default:  // instructions…;
}
```

```
Condition ? instruction1: instruction2 ;
Exemple:     $a= (2==2) ? “Oui”: “Non”;
```

Les boucles: for et foreach

for(valeur initiale; codition; modification de valeur){

//instructions …

}

exemple:

```
Echo “<ul>
for($i=0; $i<10;$i++){
        echo “<li>$i</li>
}
Echo “</ul>”
```

foreach($tableau as $valeur){

//instructions …

}

foreach($tableau as $cle=>$valeur){

//instructions …

}

Exemple:

```
$jours=Array(“ludi”,”Mardi”,”Mercredi”,”Jeudi”)
Echo “<ul>
foreach($jours  as $j){echo “<li>$j</li>}
Echo “</ul>”
```

Les boucles:while et do...while

```
while( codition est true){
        //instructions …
}
```

```
do{
        //instructions …
}while(condition est true);
```

Les fonctions

```
function maFonction($arg1,$arg2=12){

        $c=$arg1+$arg2

        //instructions

        return $c

}
```

PHP Superglobals

- Les superglobals sont des variables prédéfinie dans PHP et sont accessible dans tous les scriptes du projet:
  - ***$GLOBALS***
  - ***$_SERVER***
  - ***$_GET***
  - ***$_POST***
  - ***$_SESSION***
  - ***$_COOKIE***
  - ***$_REQUEST***
  - ***$_ENV***
  - ***$_FILES***

https://www.php.net/manual/en/language.variables.superglobals.php

PHP Superglobals: $GLOBALS

$GLOBALS: Un tableau associatif pour référencer toutes les variable dans le scop globale du scripte en cours

```
<?php
        function test() {
                $foo = "local variable";
                echo '$foo in global scope: ' . $GLOBALS["foo"] . "\n";     # Affiche:  $foo in global scope: Example content
                echo '$foo in current scope: ' . $foo . "\n";               # Affiche :  $foo in current scope: local variable
    }
        $foo = "Example content";
        test();
?>
```

PHP Superglobals: $_SERVER

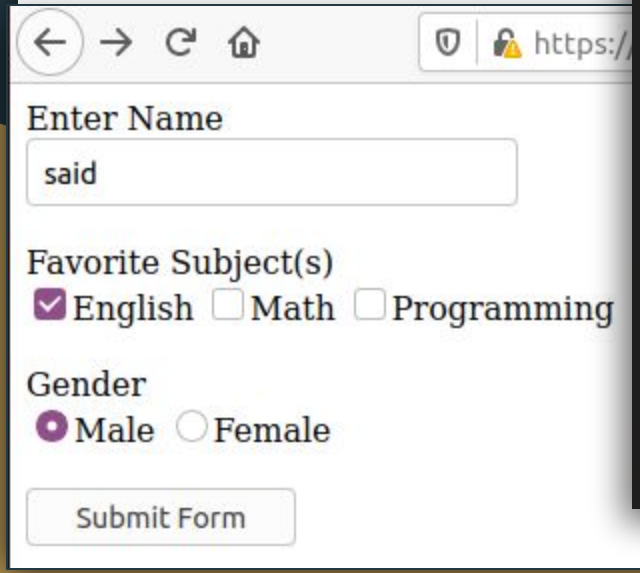
$_SERVER: Un tableau associatif contenant des information sur l'environnement d'exécution du scripte

- ***$_SERVER['PHP_SELF']:*** le chemin du scripte en cours d'exécution par exemple si le script est ***http://example.com/foo/bar.php*** alors on obtient ***/foo/bar.php***
- ***$_SERVER['SERVER_ADDR']:*** L'adresse IP du serveur
- ***$_SERVER['REMOTE_ADDR']:*** L'adresse IP du client
- ***$_SERVER['SERVER_NAME'']:*** Le nom du serveur
- ***$_SERVER['REQUEST_METHOD']:*** La méthode utilisé pour accéder à la page: GET, POST …
- ...

PHP Superglobals: $_GET, $_POST

htdocs/web/01-superglobals/form.html

```
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <title>PHP Form Handling</title>
5    </head>
6    <body>
7      <form action="process.php" method="get">
8        Enter Name <br />
9        <input type="text" name="studentname" value="Your Name" />
10       <br /><br />Favorite Subject(s)<br />
11       <input type="checkbox" name="subj[]" value="EL" />English
12       <input type="checkbox" name="subj[]" value="MA" />Math
13       <input
14         type="checkbox"
15         name="subj[]"
16         value="PG"
17       />Programming<br /><br />Gender <br />
18       <input type="radio" name="gender" value="M" />Male
19       <input type="radio" name="gender" value="F" />Female<br /><br />
20       <input type="submit" name="sm" value="Submit Form" />
21     </form>
22   </body>
23 </html>
```



PHP Superglobals: $_GET, $_POST

- Un clique sur le bouton submit permet d'envoyer les données à l'aide plusieurs méthode:
  - GET: les données sont joint à la l'URL sous le nom de "**query string**": ***https://localhost/web/01-superglobals/***process.php***?studentname=said&subj%5B%5D=EL&subj%5B%5D=PG&gender=M&sm=Submit+Form***
  - POST: Les données sont envoyé dans le body de la requête
  - ...
- Pour récupérer les données du formulaire nous avons recours aux superglobals:
  - ***$_GET["studentname"] : "said"***
  - ***$_GET["subj"] : array("EL", "PG")***
  - ***$_GET["gender"]: "M"***
  - ***$_GET["sm"]: "Submit Form"***
  - Si la méthode et POST on utilise ***$_POST[]***

PHP Superglobals: $_SESSION

- $_SESSION est un tableau associatif utilisé pour partager des données entre plusieurs pages

htdocs/web/01-superglobals/session1.php

```php
<?php
    session_start();
    $_SESSION['myFavFood'] = 'Couscous';
    $_SESSION['myFavDrink'] = 'Cola';
    $_SESSION['myFavColor'] = 'Orange';
    $_SESSION['myFavDrink'] = 'Tea'; #updating a session variable
    unset($_SESSION['myFavColor']);#deleting a session variable
?>
```

htdocs/web/01-superglobals/session2.php

```php
<?php
session_start();
echo '<BR>Food: '.$_SESSION['myFavFood']; # affiche: Food: Couscous
echo '<BR>Drink: '.$_SESSION['myFavDrink']; # affiche:  Drink: Tea
echo 'Color: '.$_SESSION['myFavColor']; # Warning: Undefined array key "myFavColor"
?>
```

PHP Superglobals: $_COOKIE

1. Un cookie est souvent utilisé pour identifier un utilisateur.
2. Un cookie est un petit fichier que le serveur embarque sur l'ordinateur de l'utilisateur.
3. Chaque fois que le même ordinateur demande une page avec un navigateur, il enverra également le cookie.
4. Avec PHP, on peut à la fois créer et récupérer des valeurs de cookies.

Créer des cookies

Un cookie est créé avec la fonction ***setcookie()***.

setcookie(name, value, expire, path, domain, secure, httponly);

Exemple:

```
<?php
        $name = "user";
        $value = "Said Amayno";
        setcookie($cookie_name, $cookie_value, time() + (86400 * 30), "/"); // 86400 = 1 day
?>
```

Utiliser des cookies

Pour accéder à un cookie on utilise le tableau associative **$_COOKIE**

Exemple:

```
<html> <body>  <?php
        if(!isset($_COOKIE[$name])) {
                echo "Cookie named '" . $name . "' is not set!"; }
        else {
                echo "Cookie '" . $name . "' is set!<br>";   echo "Value is: " . $_COOKIE[$name]; }
?>  </body> </html>
```

Supprimer un cookie

Pour supprimer un cookie, on utilise la fonction ***setcookie()*** avec une date d'expiration dans le passé :

Exemple:

```
<?php
    // set the expiration date to one hour ago
    setcookie("user", "", time() - 3600);
?>
```

Les filtres PHP

Les filtres PHP sont utilisés pour valider et nettoyer les entrées externes.

Nous avons deux types de filtres:

1. Filtres de Validation des données: Détermine si les données sont sous une forme appropriée.
2. Filtres de désinfection des données: Supprimer tout caractère illégal dans les données.

La fonction ***filter_list()*** peut être utilisée pour lister ce que propose l'extension de filtre PHP :

```
<?php

    foreach (filter_list() as $filter) {echo $filter . '<br>' }

?>
```

La fonction filter_var()

La fonction ***filter_var()*** valide et désinfecte les données.

Syntax: ***filter_var($var, FILTRE)***

- $var :La variable que vous voulez vérifier
- FILTRE: Le type de filtre à utiliser:
  - FILTER_SANITIZE_STRING
  - FILTER_VALIDATE_INT
  - FILTER_VALIDATE_IP
  - FILTER_SANITIZE_EMAIL
  - FILTER_SANITIZE_URL
  - FILTER_VALIDATE_URL

Les calbacks

1. Un calback est une fonction qui est passée en argument à une autre fonction.
2. Toute fonction PHP existante peut être utilisée comme un calback.

Exemple:

```
<?php

    function my_callback($item) {return strlen($item);}

    $strings = ["apple", "orange", "banana", "coconut"];

    $lengths = array_map("my_callback", $strings);

    print_r($lengths);

?>
```

Callback définie par l'utilisateur

```php
<?php
        function exclaim($str) {return $str . "! ";}
        function ask($str) {return $str . "? ";}
        function printFormatted($str, $format) {
                echo $format($str);
        }
        printFormatted("Hello world", "exclaim");
        printFormatted("Hello world", "ask");
?>
```

PHP et JSON

1. JSON signifie JavaScript Object Notation et est une syntaxe pour le stockage et l'échange de données.
2. PHP a des fonctions intégrées pour gérer JSON.
    - json_encode() : permet d'encoder une valeur au format JSON.
    - json_decode() : permet de décoder un objet JSON en un objet PHP ou un tableau associatif.

PHP - json_encode()

Encoder un tableau indexé dans un objet JSON

```
<?php
        $data = array( "Reda", "Hanane", "Mohammed" );
        echo json_encode($data);
?>
```

Encoder un tableau associatif dans un objet JSON

```
<?php
        $data = array( "Reda"=>15, "Hanane"=>20, "Mohammed"=>19 );
        echo json_encode($data);
?>
```

PHP - json_decode()

décode les données JSON dans un ***objet PHP***

```
<?php
        $jsonobj = '{"Reda":15, "Hanane":20, "Mohammed":19}';
        $obj=json_decode($jsonobj);
        var_dump($obj);
        echo $obj->Reda,$obj->Hanane,$obj->Mohammed;
        foreach($obj as $key => $value) {
                echo $key . " => " . $value . "<br>";
        }
?>
```

décode les données JSON dans un ***tableau associatif***

```
<?php
        $jsonobj = '{"Reda":15, "Hanane":20, "Mohammed":19}';
        $tab=json_decode($jsonobj, true);
        var_dump($tab);
        echo $tab["Reda"],$tab["Hanane"],$tab["Mohammed"];
        foreach($tab as $key => $value) {
                echo $key . " => " . $value . "<br>";
        }
?>
```

PHP -Exceptions

1. Une exception est un objet qui décrit une erreur ou un comportement inattendu d'un script PHP.
2. Des exceptions sont levées par de nombreuses fonctions et classes PHP.
3. Les fonctions et classes définies par l'utilisateur peuvent également lever des exceptions.
4. Les exceptions sont un bon moyen d'arrêter une fonction lorsqu'elle rencontre des données qu'elle ne peut pas utiliser.

Lever une exception: throw

1. L’instruction ***throw*** permet à une fonction ou méthode définie par l'utilisateur de lever une exception.
2. Lorsqu'une exception est levée, le code qui la suit ne sera pas exécuté.
3. Si une exception n'est pas interceptée, une erreur fatale se produira avec un message "Uncaught Exception".

```php
<?php

function divide($a, $b) {

    if($b == 0) {

        throw new Exception("Division by zero");

    }

     return $a / $b;

}

echo divide(5, 0);

?>// Fatal error: Uncaught Exception
```

Intercepter une exception: try...catch

Pour éviter l'erreur de l'exemple précédent, nous pouvons utiliser ***try...catch*** pour intercepter les exceptions et poursuivre le processus.

Syntax:

```
try {

        //code that can throw exceptions

        }catch(Exception $e) {

        //code that runs when an exception is caught

        }
```

```
<?php
function divide($a, $b) {
    if($b == 0) {
        throw new Exception("Division by zero");
    }
     return $a / $b;
 }
Try{
        echo divide(5, 0);
}catch(Exception $e) {
        echo "Division impossible ";
 }
?>

//Division impossible
```

Intercepter une exception: try...catch...finally

1. ***try...catch...finally*** peut être utilisée pour intercepter des exceptions.
2. Le code du bloc ***finally*** s'exécutera toujours, qu'une exception ait été interceptée ou non. Si ***finally*** est présent, le catch ***bloc*** est facultatif.

Syntax:

try {

//code that can throw exceptions} ***catch(Exception $e) {***

//code that runs when an exception is caught} ***finally {***

//code that always runs regardless of whether an exception was caught

}

```
<?php
function divide($a, $b) {
    if($b == 0) {
        throw new Exception("Division by zero");
    }
     return $a / $b;
 }
Try{
        echo divide(5, 0);
}catch(Exception $e) {
        echo "Division impossible ";
 }finally{
        echo “Fin du calcul”;
}
?>
//Division impossible
//Fin du calcul
```

Les fichiers

1. La gestion des fichiers est une partie importante de toute application Web.
2. PHP a plusieurs fonctions pour créer, lire, télécharger et éditer des fichiers.
    - ***readfile()***: lire le contenu d'un fichier
    - ***fopen()***: ouvrir un fichier existant ou créer un nouveau
    - ***fread()***: lire le contenu d'un fichier
    - ***fclose()***:
    - ***fgets():***
    - ***feof():***
    - ***fgetc():***

https://www.w3schools.com/php/php_ref_filesystem.asp

readfile()

1. La fonction ***readfile()*** lit un fichier et l'écrit dans le tampon de sortie.
2. Cette fonction est utile si tout ce que vous voulez faire est d'ouvrir un fichier et de lire son contenu.

Exemple:

```
<?php

        $txt= readfile("data.txt");

        echo $txt

?>
```

fopen()

1. ***fopen()*** est meilleure que ***fread()*** pour ouvrir des fichiers.
2. Cette fonction offre plus d'options que la fonction ***readfile()***.

Syntax:

fopen($filename, mode)

- ***mode:***
  - ***'r'*** *: Ouverture en* ***lecture*** *seule*
  - ***'w'****: Ouverture en* ***écriture*** *seule avec* ***suppression*** *du contenu originale du fichier*
  - ***'a':*** *Ouverture en* ***écriture*** *seule avec* ***conservation*** *du contenu originale du fichier*
  - ***'x': Créer*** *un nouveau fichier en écriture seul si ce fichier n'existe pas sinon erreur*
  - ***'r+'*** *: Ouverture en* ***lecture/écriture*** *(* ***pointeur au début de fichier*** *)*
  - ***'w+'****: Ouverture en* ***lecture/écriture efface*** *le contenu du fichier et créer le fichier s'il n'existe pas (* ***pointeur au début de fichier*** *)*
  - ***'a+':*** *Ouverture en* ***lecture/écriture*** *(* ***pointeur à la fin de fichier*** *) ajout du contenu au fichier*
  - ***'x+': Créer*** *un nouveau fichier en* ***lecture/écriture*** *si ce fichier n'existe pas sinon erreur*

fread()

1. La fonction ***fread()*** lit à partir d'un fichier ouvert.
2. Le ***premier paramètre*** de ***fread()*** contient le **nom du fichier** à lire et
3. le ***deuxième paramètre*** spécifie le ***nombre maximal d'octets à lire***.

Le code PHP suivant lit le fichier "data.txt" jusqu'à la fin :

```
<?php

        $myfile=fopen("data.txt",”r”);

        $str=fread($myfile,filesize("data.txt"));

        echo $str

        fclose($myfile) //fermer le fichier ouvert

?>
```

fgets()

fgets() est utilisée pour lire une seule ligne d'un fichier.

Exemple:

Lire la première ligne du fichier "data.txt" :

```
<?php
        $myfile = fopen("data.txt", "r") or die("Unable to open file!");
        $line=fgets($myfile);
        echo $line
        fclose($myfile);
?>
```

feof()

1. feof() vérifie si la "fin de fichier" (EOF) a été atteinte.
2. feof() est utile pour parcourir des données de longueur inconnue.

Exemple:

Lire le fichier "data.txt" ligne par ligne, jusqu'à ce que la fin du fichier soit atteinte

```
<?php
        $myfile = fopen("data.txt", "r") or die("Unable to open file!");
        while(!feof($myfile)) {
                echo fgets($myfile) . "<br>";
        }
        fclose($myfile);
?>
```

fgetc()

fgetc() est utilisée pour lire un seul caractère d'un fichier.

Exemple: Lire le fichier "data.txt" caractère par caractère, jusqu'à ce que la fin du fichier soit atteinte .

```
<?php
        $myfile = fopen("data.txt", "r") or die("Unable to open file!");
        while(!feof($myfile)) {
                echo fgetc($myfile).”<br/>”
        }
        fclose($myfile);
?>
```