



Bachelier en Informatique de Gestion

Projet de Développement Web

Enseignement supérieur économique de type court

Code FWB : 7534 30 U32 D3

Code ISFCE : 4IPW3



= BUROTIX ()

Table des matières

Généralités

- 01. Introduction au web
- 03. Outils
- 05. Frameworks

Côté Client

- 12. Structure HTML
- 13. Formulaire HTML
- 14. Mise en forme CSS
- 15. Adaptabilité
- 17. Javascript
- 18. Framework jQuery
- 19. AJAX

Côté Serveur

- 21. Middleware PHP
- 22. Traitement du formulaire
- 23. Architecture MVC
- 24. Base de données SQL
- 25. Données XML
- 26. Données JSON





21. MiddlewarePHP

PHP en HTML

Manipulation de fichiers

Bases du langage PHP

Manipulation de tableaux



= BUROTIX ()

Propos liminaire : Pourquoi PHP ?

- PHP est le langage utilisé pour 75% des sites web au niveau mondial.
 - Remarque : Notamment avec WordPress
 - DOT NET : 6%
 - Java : 5%
 - Javascript : 4%
- Source
 - W3TECHS, septembre 2024





PHP en HTML



= BUROTIX ()

PHP en HTML

- Balises PHP
 - Le code PHP s'insère au milieu du code HTML au sein d'une balise `<?php . . . ?>`
 - Ces bouts de code seront la partie dynamique du site web.
 - On ne trouve pas de fonction `"main()"` en PHP comme on en trouve dans d'autres langages (C, etc.)
 - Les scripts PHP sont utilisés pour écrire du code HTML de manière dynamique.
- Fichiers PHP
 - Les fichiers contenant du code PHP doivent avoir l'extension `.php`.
 - Les fichiers d'extension `.htm` ou similaire ne seront pas interprétés comme php.
 - C'est uniquement le serveur web qui interprête (exécute) le code PHP, jamais le navigateur.

PHP en HTML

Écriture d'un texte. Exemple.

Le serveur web/php qui lit ce code mixte PHP/HTML

```
<p>  
    Bonjour ceci est un texte</br>  
    <?php echo "Bienvenue sur mon site"; ?>  
</p>
```

va envoyer au client cette page HTML

```
<p>  
    Bonjour ceci est un texte</br>  
    Bienvenue sur mon site  
</p>
```



PHP en HTML

PHP est également utilisé pour générer des balises HTML. Exemple.

Le serveur web/php qui lit ce code mixte PHP/HTML

`<p>`

Bonjour ceci est un texte`</br>`

`<?php echo "<h1>Bienvenue sur mon site</h1>"; ?>`

`</p>`

va envoyer au client cette page HTML

`<p>`

Bonjour ceci est un texte`</br>`

`<h1>`Bienvenue sur mon site`</h1>`

`</p>`



= BUROTIX ()

PHP en HTML

Des commentaires peuvent être ajoutés à tous les scripts PHP. Exemple.

```
<p>
    Bonjour ceci est un texte</br>
    <?php
    /* Ceci est un commentaire et n'apparaîtra pas
       après interprétation du script PHP */
    echo "<h1>Bienvenue sur mon site</h1>";
    ?>
</p>
=>
<p>
    Bonjour ceci est un texte</br>
    <h1>Bienvenue sur mon site</h1>
</p>
```





Bases du langage



= BUROTIX ()

Début, fin, instructions, commentaires

- Les scripts (programmes) démarrent toujours par les 2 symboles suivants :

```
<?php
```

```
...
```

```
...
```

```
?>
```

- Commentaires :

```
<?php
```

```
...
```

```
/*
```

```
 * Ceci est commentaire sur
```

```
 * plusieurs lignes
```

```
 */
```

```
...
```

```
// Ceci est commentaire sur  
une seule ligne
```

```
...
```

```
?>
```

- Les instructions se terminent TOUJOURS par un point-virgule

```
<?php
```

```
...
```

```
instruction1;
```

```
instruction2;
```

```
...
```

```
?>
```



Messages d'erreur

- PHP vous aide à détecter les erreurs dans votre code en écrivant des messages d'erreur. Lisez-les attentivement:

```
<?php  
echo "Bonjour"  
echo "Aurevoir";  
?>
```

```
Parse error: syntax error, unexpected 'echo' (T_ECHO) ,  
expecting ',' or ';' in main.php on line 3
```



Nom de variable

- Un nom de variable peut comporter des lettres, des chiffres et le caractère `_` (les espaces ne sont pas autorisés !)
- Un nom de variable doit commencer par un `$` suivi d'une lettre ou du caractère `_`
- Un nom de variable peut comporter des lettres, des chiffres et le caractère `_` (les espaces ne sont pas autorisés!)
- Les noms de variable sont sensibles à la casse (différence entre minuscule et majuscule)

- Correct ou non ?

`$variable`

`$Nom de Variable`

`$123Nom_De_Variable`

`$nom_de_variable_123`

`$nom_de_variable`

`$Variable`

`$toto@mailcity.com`

`nom_de_variable`

`$Nom-de-variable`



Types de variables scalaires

- Types numériques :

`int` ou `float`

```
$myAge = 16;
```

```
$yourAge = 15.5;
```

- Types caractères :

`char`, `string`

```
$greeting = "Hello!"
```

- Types logiques :

`bool`

```
$hasHair = true;
```

```
$hasHair = false;
```



Constantes

- Par convention, on écrira les constantes en majuscule

```
define (PI, 3.14159265) ;
```

```
define (NOM_ENTREPRISE, "Carrefour") ;
```

```
Echo "Bienvenue chez " ;
```

```
echo NOM_ENTREPRISE ;
```

- Remarque: Contrairement aux variables, il n'y a pas de \$ devant le nom des constantes



Expressions

- Affectation

```
$x = 3; // integer
```

```
$y = 3.5; // float
```

- Affectation de chaînes de caractères

```
$s = "la variable x vaut $x.";
```

```
// output : la variable x vaut 3.
```

```
$t = 'la variable x vaut $x';
```

```
// output : la variable x vaut $x.
```

- Egalité

```
$x == $x + 1
```



Expressions

- Affectation HEREDOC

```
$s = <<< EOT
    Hello, World $x!
EOT;
```

NB : `$x` est interprété

- Affectation NOWDOC

```
$s = <<< 'EEE'
    Hello, World $x!
EEE;
```

NB : `$x` n'est pas interprété

- Utilisable aussi avec **echo**

```
echo <<< EOT
    Hello, World $x!
EOT;
```



Expressions

- Ecriture à l'écran

`echo $x;`

- Retour à la ligne

`echo "\r\n";`

ou

`echo "\n";`

NB : Intérêt réduit en
HTML

- Autres fonctions
d'impression

- `printf()`

- `ob_start()`

- `ob_get_contents()` ...



Opérateurs de relation

==	égal
!=	différent
<	strictement inférieur
>	strictement supérieur
<=	inférieur ou égal
>=	supérieur ou égal

!	négation logique
&&	"et" logique
and	"et" logique
	"ou" logique
or	"ou" logique



Instruction conditionnelle

```
if( /* CONDITION */ )
{
    /* INSTRUCTIONS 1*/
}
else
{
    /* INSTRUCTIONS 2*/
}
```

```
if ($a == 1)
{
    echo 'oui, vrai';
}
else
{
    echo 'non, faux';
}
```



Opérateur ternaire

```
if ($a==1)
{
    echo 'oui';
}
else
{
    echo 'non';
}
```



```
echo $a==1 ? 'oui' : 'non';
```

```
if ($a==1)
{
    $b = 2 ;
}
else
{
    $b = 3 ;
}
```



```
$b = $a==1 ? 2 : 3 ;
```



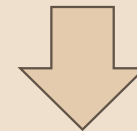
Opérateur ternaire

```
if ($a)
{
    $b = $a ;
}
else
{
    $b = -1 ;
}
```



```
$b = $a ?: -1 ;
```

```
if (isset($a))
{
    $b = $a ;
}
else
{
    $b = -1 ;
}
```



```
$b = $a ?? -1 ;
```



Opérateurs arithmétiques

$\$a + \b	Addition
$\$a - \b	Soustraction
$\$a * \b	Multiplication
$\$a / \b	Division réelle
$(int) (\$a/\$b)$ ou $intdiv(\$a, \$b)$	Division entière, $\$a$ et $\$b$ entiers soit par le "cast" (troncature des décimales) soit par la fonction dédiée $intdiv()$
$\$a \% \b	Modulo (reste de la division entière)



Ordre de priorité des opérateurs (en bref)

- Opérateurs par priorité :

1. ******
2. **!**
3. ***** **/**
4. **+** **-**
5. **<** **<=** **>** **>=**
6. **==** **!=**
7. **&&**
8. **||**
9. **and**
10. **or**

- Évolutif (version PHP) !
- Sans parenthèse : opérateurs de même priorité évalués de gauche à droite
 - **4/2*3** $\leq \Rightarrow$ **(4/2)*3**
- Avec parenthèses : ordre d'évaluation des opérateurs explicitement fixé
 - **(5+3)/2** \Rightarrow 4
 - **5+3/2** \Rightarrow 6.5

Exercice 11 : emprunt

Données	<p>IN:</p> <ul style="list-style-type: none">• dettesActuelles: integer• epargne: integer• prêtDemandé: integer
Interactions	<ul style="list-style-type: none">• Pour un client donné, demander le montant des dettes, de l'épargne et du prêt demandé• écrire "Prêt accordé" si le montant des dettes cumulées ne dépassera pas 75% de l'épargne du client sinon écrire "Prêt refusé"



Exercice 12 : mini-factory

- **DONNÉES EN ENTRÉE:**

- P : prix hors TVA de l'article acheté en €
- Q : quantité commandée de l'article en €
- T : taux de TVA à appliquer en décimal (ex.: 0.21)

- **PRE-CONDITIONS:**

- P : réel strictement positif
- Q : entier strictement positif
- T : réel compris entre 0 et 1

- **RÉSULTATS:**

- Afficher le montant à payer TVA comprise (21%)
- Si le montant HTVA est supérieur à 500€, alors compter au client une remise de 10% (sur le montant HTVA).



Boucle

- Boucle **"while"**

```
$cpt = 0 ;  
while( $cpt <= 10 )  
{  
    echo "$cpt " ;  
    $cpt++ ;  
}
```

- Boucle **"for"**

```
for( $i=0; $i<10; $i++ )  
{  
    echo "$i " ;  
}
```



Exercices 21 à 25

21. Affichez les 100 premiers nombres entiers positif à partir de 0 (inclus)
22. Affichez les nombres pairs entre 50 et 80 (inclus)
23. Affichez les multiples de 5 compris entre m et p inclus. On ne sait pas si $m < p$ ou l'inverse.
24. Calculez $m * p$ sans utiliser l'opérateur `"*"`.
25. Calculez $n!$





Fonctions

Déclaration

- Nom de fonction valide
 - 1^{er} caractère : lettre ou "_"
 - Ensuite, nombre quelconque de lettres, nombres ou "_"
 - Insensible à la casse
- Déclaration et utilisation indépendantes
 - Non nécessaire de déclarer la fonction avant de l'utiliser
- Portée globale
- Pas de surcharge
- Destruction ou redéfinition de fonctions déclarées impossible.
- Fonctions récursives autorisées
- Exemple :

```
function hello_world()  
{  
    echo "hello world";  
}
```

Argument de fonction

- Passage d'arguments
 - par valeur (défaut)
 - par référence
 - valeurs d'arguments par défaut possible
 - liste variable d'arguments possible



Arguments, passage par valeur

- Exemple

```
//declaration
function takes_array($input)
{
    echo "$input[0] + $input[1] = ", $input[0]+$input[1];
}

// appel
takes_array( array(4,5) );
```



Arguments, passage par référence

- Exemple

```
function add_some_extra( & $string )  
{  
    $string .= ', et un peu plus.';  
}
```

```
$str = 'Ceci est une chaîne';  
add_some_extra( $str ); // argument = variable !  
echo $str; // 'Ceci est une chaîne, et un peu plus.'
```



Arguments, valeurs par défaut

- Exemple

```
function servir_cafe ($type = "cappuccino")  
{  
    echo "Servir un $type.\n";  
}
```

```
servir_cafe();          // ' Servir un cappuccino.'
```

```
servir_cafe(null);     // ' Servir un .'
```

```
servir_cafe("espresso");  
                    // 'Servir un espresso.'
```



Retour de fonctions

- Retour de valeur scalaire, exemple

```
function carre($num)
{
    return $num * $num;
}
```

```
$x = carre(4);
echo carre(4);
// Affiche '16'
```

- Retour de valeur complexe, exemple

```
function pn()
{
    return array(0, 1, 2);
}

list($ze, $un, $de) = pn();
```

- Remarque : Déclaration des types de retour possible en PHP 7



Fonctions internes

- Une seule référence
 - <https://www.php.net/manual/fr/funcref.php>
- Pour connaître les bibliothèques et extensions chargées sur votre serveur
 - Fonction `phpinfo()`
 - Fonction `get_loaded_extensions()`
 - Serveur WAMP : `localhost` vous donne cette information



Fonction anonyme

- Les fonctions anonymes (ou *closure*) permettent la création de fonctions sans préciser leur nom.
- Usage spécifique : les *callback*
- Exemple

```
$greet = function($name)
{
    printf("Bonjour %s\r\n", $name);
};
```

```
$greet('World');
$greet('PHP');
```



Typage des arguments de fonction (> PHP 7)

- Par défaut, PHP convertit le type d'origine vers le type scalaire attendu.
- Le **typage** permet à une fonction de requérir qu'un paramètre soit d'un certain type lors de l'appel de celle-ci.
 - PHP 5 : typage des classes et arrays
 - PHP 7 : typage des scalaires également
- Activation possible d'un **typage strict**
 - Depuis PHP 7
 - Fichier par fichier
 - Uniquement pour les types scalaires
- Principe
 - Seule une variable du type exact correspondant au type attendu dans la déclaration sera acceptée
 - Sinon exception **TypeError** levée
 - Expression **declare** utilisée avec **strict_types**



Typage : exemple

```
declare(strict_types=1) ;
```

```
function sum(int $a, int $b)  
{  
    return $a + $b;  
}
```

```
echo sum( 1, 2 ) ; // OK
```

```
echo sum( 1.5, 2.5 ) ; // KO
```





Manipulation de tableau



= BUROTIX ()

Tableau indexé

- Éléments de même type
- Exemple : une série de noms

```
$nomEtudiant = ["Julie", "Sophie", "Maxime"]; // PHP 7  
$nomEtudiant = array( "Julie", "Sophie", "Maxime" );
```

- Index courant à partir de 0

```
echo $nomEtudiant[1]; // "Sophie"
```

- Taille du tableau

```
$size = count($nomEtudiant);
```

- Ajouter un élément au tableau

```
$nomEtudiant[] = "Kevin";
```



Tableau indexé : Boucle

- Boucle **foreach**
- Ni index, ni initialisation, ni incrément, ni condition
foreach(**\$tab** **as** **\$value**)
{
 // instructions
}
- Les boucles **for** et **while** restent applicables aux tableaux.



Tableau indexé : Boucle

- Exemple 1:

```
$tab = [ "Julie", "Sophie", "Maxime" ];  
foreach($tab as $nom)  
{  
    echo $nom . "<br>";  
}
```

- Exemple 2 :

```
foreach( array(5,6,2,4,5,9,6) as $val )  
{  
    echo <<< HTML  
        <div style="background-color:#$val$val$val;">  
            $val  
        </div>;  
    HTML;  
}
```



Exercices 31 à 33

Remarque : Par "tableau" il est entendu "tableau de nombre".

31. Concevez un algorithme qui calcule la moyenne d'un tableau.

32. Additionner les éléments de deux tableaux de même taille pour former un troisième tableau, et

multipliez-les pour former un quatrième tableau.

33. Concevez un algorithme qui trouve le maximum et le minimum d'un tableau.

Tableau associatif

- Remarque : appelé "*dictionnaire*" dans d'autres langages
- Usage 1: Similaire au tableau indexé, index remplacé par **clé**
- Exemple : la liste des salaires d'une entreprise

```
$salaries = array(  
    "mohammad" => 2000,  
    "qadir" => 1000,  
    "zara" => 500  
);
```



Tableau associatif

- Accès en lecture

```
$x = $salaries['qadir'];
```

- Accès en écriture

```
$salaries['hammad'] = 3500 ;
```

- Destruction

```
unset($salaries['zara']);
```



Tableau associatif : Boucle

- Boucle **foreach**

```
foreach( $tab as $key => $value )  
{  
    // instructions  
}
```



Tableau associatif : Boucle

```
$salaries = array(  
    "mohammad" => 2000,  
    "qadir" => 1000,  
    "zara" => 500  
);  
foreach($salaries as $nom => $sal)  
{  
    echo "$nom gagne $sal euro";  
}
```



Tableau associatif

- Usage 2 : Éléments de types hétérogènes
- Exemple : un enregistrement d'une table de BD

```
$member = array(  
    "name"    => "Peter Parker",  
    "email"   => "peterparker@mail.com",  
    "salary" => 2000,  
);
```

- Accès : similaire



Tableau multidimensionnel

Etudiant	Cours									
	0	1	2	3	4	5	6	7	8	9
0	10	13	15	19	16	18	13	13	12	13
1	12	3	5	9	10	12	3	10	12	10
2	12	15	14	19	16	20	15	16	13	13
3	10	16	12	14	15	17	13	14	15	7
4	16	12	16	12	10	11	9	14	13	9

- Usage 1 : Tableau indexé de tableaux indexés
- Exemple : notes d'une classe d'étudiants

```
$noteEtudiant = [  
    [ 10,13,15,19,16,18,13,13,12,13 ],  
    [ 12, 3, 5, 9,10,12, 3,10,12,10 ],  
    [ 12,15,14,19,16,20,15,16,13,13 ],  
    [ 10,16,12,14,15,17,13,14,15, 7 ],  
    [ 16,12,16,12,10,11, 9,14,13, 9 ]  
];
```

- Accès : `$noteEtudiant[2][5] // 20`

Tableau multidimensionnel

- Usage 2 : Tableau indexé de tableaux associatifs
- Exemple : une requête dans une base de données

```
$contacts = array(  
    array(  
        "name" => "Peter Parker",  
        "email" => "peterparker@mail.com",  
    ),  
    array(  
        "name" => "Clark Kent",  
        "email" => "clarkkent@mail.com",  
    ),  
    array(  
        "name" => "Harry Potter",  
        "email" => "harrypotter@mail.com",  
    ),  
);
```

- Accès : `$contacts[1]['email'] // "clarkkent@mail.com"`



Exercices 36 : Facture

- Input
 - Tableau indexé de 3 tableaux associatifs
 - Par tableau associatif
 - Taux de TVA
 - Prix unitaire
 - Quantité
- Processing
 - Sous-totaux HT
 - Montant TVA
 - Sous-totaux TTC
 - Totaux HT et TTC
- Output
 - facture avec trois sous-totaux et les totaux HT et TTC (cf screenshot).

PU	QNT	P HT	TVA	P TTC
50	20	1000	210	1210
35	3	105	22	127
5	1	5	1	6
TOTAL HT		:	1110	
TOTAL TVAC		:	1343	





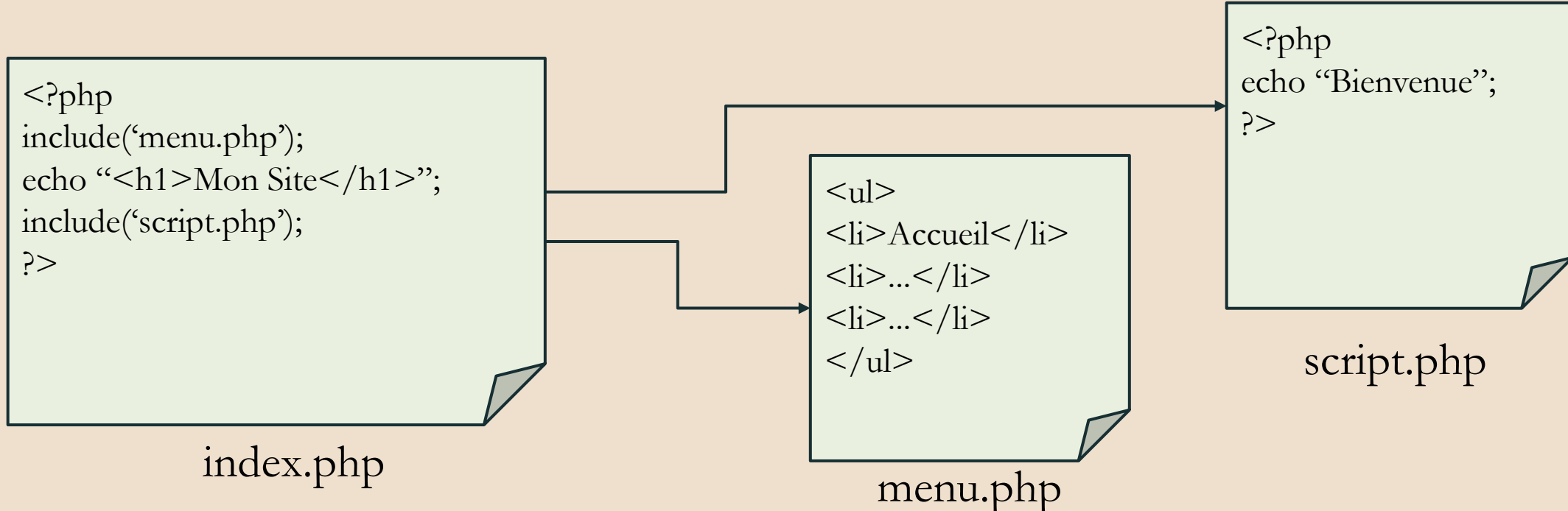
Manipulation de fichiers



= BUROTIX ()

Inclusion de code PHP / HTML

- **include(' fileName ')**
 - appel à un script **.php** depuis une autre page **.php**
 - code de la page incluse exécuté au moment de l'inclusion



Inclusion de code PHP / HTML

- La famille de fonctions **include** peut être utilisée pour:
 - insérer du code HTML
 - insérer un code PHP à exécuter
 - insérer un ensemble de fonctions PHP
 - organiser un développement en équipe
 - ...
- Variantes :
 - **include**, **include_once**, **require**, **require_once**
 - Variante recommandée : **require_once**



Accès à des fichiers de données

- **fopen**
 - ouvrir un fichier (texte)
- **fclose**
 - fermer un fichier
- **fgets**
 - lire une ligne de texte et avancer le pointeur à la ligne suivante
- **feof**
 - Teste si un pointer se situe à la fin du fichier
- **explode**
 - découper une chaîne de caractères en tableau
- **implode**
 - construire une chaîne de caractères à partir d'un tableau
- **fwrite**
 - écrire un string dans un fichier

fopen

- **fopen(\$fileName, \$mode);**
 - ouvrir un fichier (texte)
 - **\$fileName**
 - le chemin du fichier sur le serveur
 - **\$mode**
 - **r**: ouvrir le fichier en lecture
 - **w**: ouvrir le fichier en écriture (pointer au début du fichier, le reste est supprimé)

- **a**: ouvrir le fichier en écriture (pointer à la fin du fichier)

- **return**

- “ressource” ou “file handler” ou “pointeur de fichier”

- Exemple:

- **\$handle = fopen("file.txt", "r");**



fclose

- `fclose($filePointer);`

- fermer un fichier

- `$filePointer`

- indique le chemin du fichier

- Exemple

```
$handle = fopen("/home/rasmus/file.txt", "r");
```

```
fclose($handle);
```



fgets

- **`fgets ($filePointer) ;`**
 - lire une ligne de texte et avancer le pointeur à la ligne suivante
 - **`$filePointer`**
 - Ressource fichier
 - **`return`**
 - String (la ligne lue)
- Exemple

```
$handle = fopen("/home/rasmus/file.txt", "r");  
$ligne = fgets($handle);  
fclose($handle);
```



feof

- **feof(\$filePointer) ;**

- Teste si un pointer se situe à la fin du fichier

- **\$filePointer**

- Ressource fichier sur lequel on teste le “End-Of-File”

- **return**

- boolean

- Exemple

```
$handle=fopen("file.txt","r") ;  
while( ! feof($handle) ) {  
    $ligne=fgets($handle) ;  
    // process $ligne ...  
}  
fclose($handle) ;
```



explode

- **explode(\$delimiter, \$string);**
 - découper une chaîne de caractères suivant un délimiteur en retournant un tableau contenant chaque élément
 - **\$delimiter**
 - indique la chaîne de caractère à utiliser comme délimiteur
 - **\$string**
 - chaîne à découper
 - **return:** array
- Exemple

```
$str = "Laurent%5%iPhone%22/01/2014";  
$tab = explode("%", $str);  
$qty = $tab[1];
```
- La fonction inverse **implode()** existe aussi.



fwrite

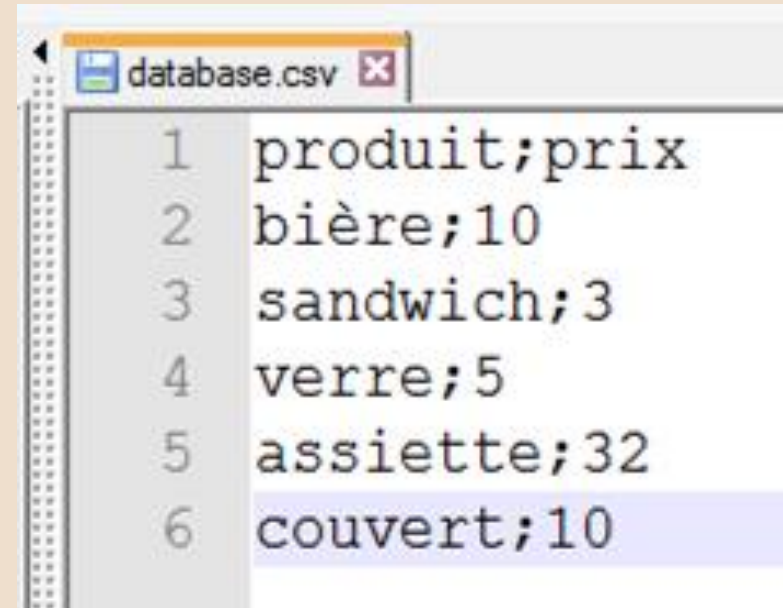
- **fwrite(\$filePointer, \$string);**
 - écrit le contenu de la chaîne string dans le fichier pointé par handle.
 - **\$filePointer**
 - Ressource fichier
 - **\$string**
 - Chaîne de caractère à insérer
- Exemple

```
$handle = fopen("/home/rasmus/file.txt", "w");  
$string = "Hello World!";  
fwrite($handle, $string);
```
- Alias de **fputs()**



Exercice 41 : Catalogue

- Afficher un tableau HTML à partir de données CSV
 - Balise TABLE ...
 - Balise LABEL DIV ...
- Ci-contre un exemple de fichier CSV et de présentation HTML.



```
1 produit;prix
2 bière;10
3 sandwich;3
4 verre;5
5 assiette;32
6 couvert;10
```

Catalogue	
produit	prix
bière	10
sandwich	3
verre	5
assiette	32

Exercice 42 : Facture

PU	QNT	P HT	TVA	P TTC
50	20	1000	210	1210
35	3	105	22	127
5	1	5	1	6
TOTAL HT		:	1110	
TOTAL TVAC		:	1343	

- Input
 - Fichier CSV (à créer) contenant les informations de base de la facture
 - Par ligne de facture :
 - Taux de TVA
 - Prix unitaire
 - Quantité
- Processing
- Lire le fichier et construire les tableaux associatifs
- Calculer les sous-totaux
- Calculer les totaux
- Output
 - facture avec sous-totaux et totaux TTC (cf screenshot).