Licence TAIS

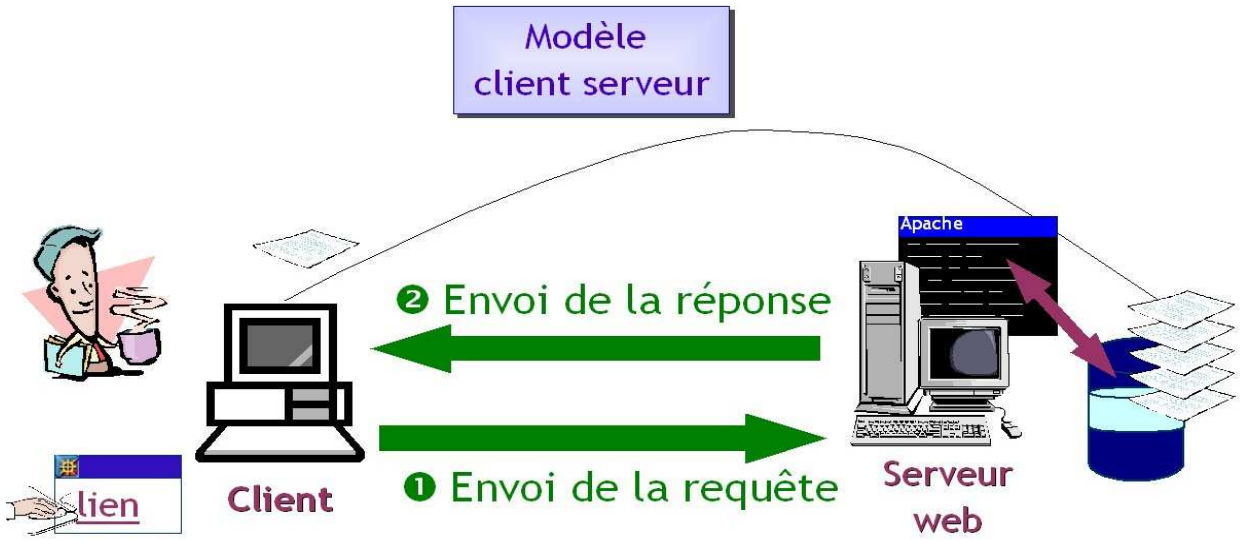
Module UE2 - M213 - PHP et MySQL

Partie 1 - Les Bases

Nous avons vu :

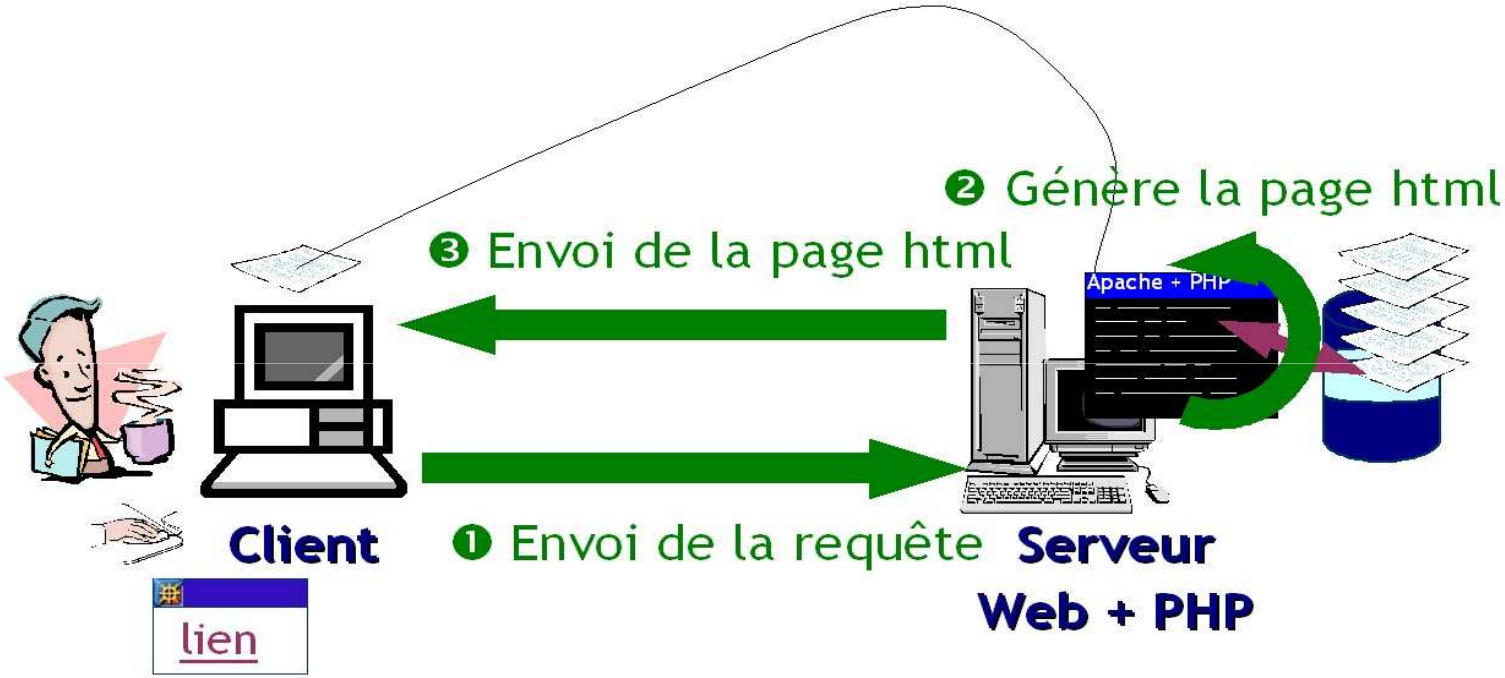
- Architecture Client/serveur (2-tiers)
  - Client et Serveur pour le Web.
    - Le client est un navigateur (Internet Explorer, Firefox, Opera, Chrome…).
    - Les site Web sont hébergés sur des serveurs dédiés nommés serveurs Web (Apache, IIS, …)

Page html avec un serveur web



Introduction à PHP

Modèle client serveur web avec moteur PHP



PHP est l'acronyme récursif de « Hypertext Preprocessor » ou « le préprocesseur hypertexte ». Ce nom traduit le fait que PHP a été conçu spécifiquement pour le traitement des pages html.

Introduction à PHP

- Présentation PHP:
  - PHP est un langage de programmation. C’est parce qu'il est gratuit, simple, facile à apprendre et puissant qu’il est connu et répandu. PHP sert surtout à écrire des sites web dynamiques.
  - Un site web dynamique est un site dont le contenu change ou évolue en fonction des actions du visiteur. Exemples : un moteur de recherche, un site marchand, un forum …

Environnement de travail

- Installation d'**EasyPHP** un outil « tout en un »
  - Logiciel entièrement gratuit qui installe un environnement de test PHP et MySQL.
    - Apache (Serveur Web)
    - PHP (Le langage de script libre)
    - Base de données MySQL (serveur de Bases de Données relationnelles Open Source)
    - phpMyAdmin (Outil graphique permettant de gérer des BD MySQL)

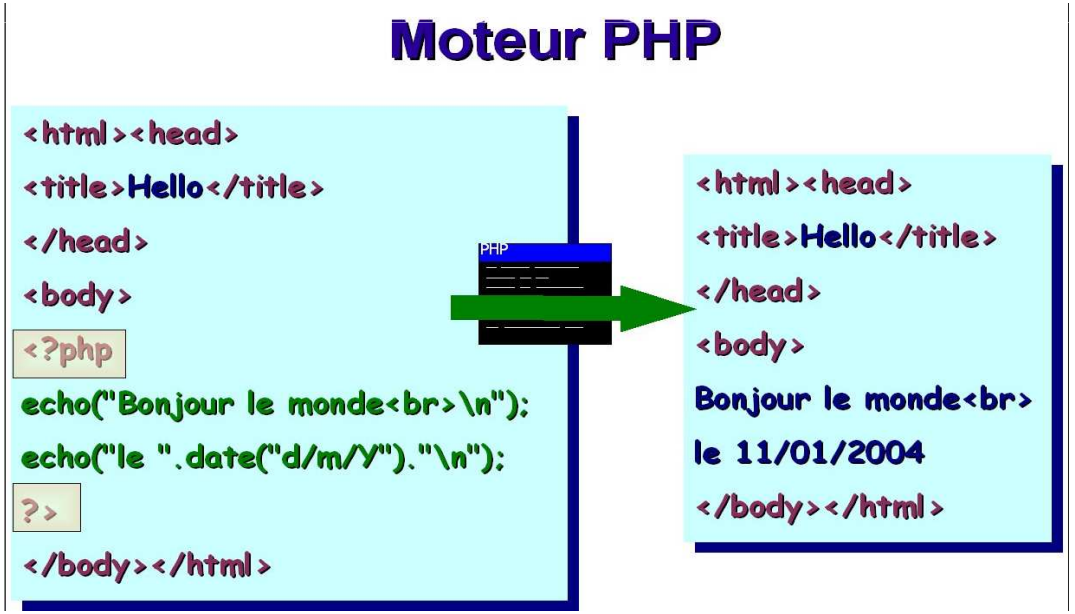
Environnement de travail

- Installation d'**EasyPHP** un outil « tout en un »
  - Démarrez le programme EasyPHP

  - Vérifiez que tout est bien installé et configuré
    - Créez dans le dossier de votre site Web (c:\Program files\EasyPHP1-8\www) un fichier appelé phpinfo.php contenant la ligne : <? Php phpinfo(); ?>
    - Ouvrez votre navigateur sur l'URL : http://localhost/phpinfo.php
    - Accédez également à l'URL : http://localhost/mysql/

PHP : les bases

- Un script PHP est composé d'un ou plusieurs blocs de code PHP marqués par des balises spécifiques *<?php … code … ?>*, et éventuellement de code HTML.
- Le code PHP est composé d'instructions terminée par ;
- Le script porte l'extension .php
- Le client reçoit une page dans laquelle il n'y a aucune instruction PHP.



PHP : les bases

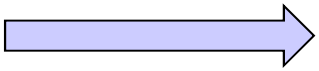
- Blocs : des portions de code PHP peuvent apparaître entre accolade, ce sont des *blocs*.
- Commentaires :
  - // indique que le texte jusqu'à la fin de la ligne doit être ignoré.
  - /* … */ permet de commenter plusieurs lignes
- Envoi de données vers la sortie standard
  - print arg; envoie l'argument arg sur la sortie standard
  - echo arg_1,arg_2, … , arg_N envoie un nombre quelconque d'arguments vers la sortie standard.

```
<?php
        print "Hello World";
        print " !";
?>
```

=

```
<?php
        echo "Hello World"," !";
?>
```

Script PHP (serveur)



Hello World !

Page HTML (client)

PHP : les bases

Exercice 1:

Ecrire le script PHP (horaire.php), pour obtenir le résultat suivant :

Code HTML résultant de l'exécution de horaire.php

```
<<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "                                    ">
<HTML>
<HEAD>
        <TITLE> Horaires de livraison</TITLE>
        <META http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
</HEAD>
<BODY>
<H1>Fastfood en ligne</H1>
<DIV>
        Il est 23:35:11 nous livrons de <EM>18:00 &agrave; 22:00</EM>
</DIV>
</BODY>
</HTML>
```

PHP : les bases

- Variables et types:
  - Toutes variables commence par "$" suivi d'un identifiant, à noter que PHP est sensible à la casse $heure_courante ≠ $Heure_courante.
  - PHP propose huit types :
    - *Integer* : entier positifs et négatifs
    - *Double* et *Float* : réels tels que 3.14, .4, $3.5^{e}12$
    - *Boolean* : deux états
      - Vrai (TRUE et tout entier non nul)
      - Faux (FALSE ou 0)
    - *String* : chaînes de caractères écrites entre guillemets ou apostrophes
    - *Array* : tableaux c'est-à-dire des collections de données
    - *Object* : objets
    - *Resource :* représente une ressource externe, certaine fonctions de manipulation de répertoires et fichiers ainsi que des fonctions MySQL
    - *Null :* comporte une unique valeur NULL qui représente l'abscence de valeur
  - Affectation d'une valeur
    - $nom_variable = expr;
    - expr peut être une entité (nombre, chaîne, variable, constante …) ou une combinaison d'entités et d'opérateurs
  - Affichage de la valeur
    - Donner comme argument à *echo* ou *print*
  - Affectation d'une adresse
    - $var1 = &$var2;
    - Une variable affectée par adresse peut être considérée comme un alias vers la variable référencée.

PHP : les bases

- Variables et types : script Valeur/adresse

```
…
        <TITLE>Valeur/adresse</TITLE>
…
<DIV>
        <?php
        echo "Affectations de valeurs<BR>";
        $x=12;
        $y=18;
        $x=$y;
        $y=5;
        echo "x=",$x," et y=",$y;
        echo "<BR>Affectation d'une adresse et de valeurs";
        $x=12;
        $y=18;
        $x=&$y;
        echo "<BR>x=",$x," et y=",$y;
        $y=5;
        echo "<BR>x=",$x," et y=",$y;
        $x=7;
        echo "<BR>x=",$x," et y=",$y;
        ?>
</DIV>
…
```

PHP : les bases

- Variables et types : script nombres.php

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
<HTML>
<HEAD>
        <TITLE>Nombres</TITLE>
        <META http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
</HEAD>
<BODY>
<DIV>
        <?php
        // Définissez les variables
        $quantité = 30;
        $prix = 119.95;
        $taxe = .05;
        //Réalisez les calculs
        $total = $quantité * $prix;
        $total = $total + ($total * $taxe);
        //formatez le total
        // La fonction number() regroupera le total en millier et l'arrondira à deux chiffres après la virgule
        $total = number_format ($total,2);
        //$total = number_format ($total,2,'.',' '); Autre type d'affichage
        //Imprimez le résultat
        echo '<p> Vous achetez <em>'.$quantité.'</em> produits au prix de <b>'.$prix.'</b> Euros. Avec la TVA le total
est de <b>'.$total.'</b> Euros.</p>';
        ?></DIV>
</BODY>
</HTML>
```

PHP : les bases

- Constantes
  - Permet de stocker entier ou réel, booléen, chaîne de caractères
  - Par convention toujours écrites en majuscules
  - L'affectation d'une valeur à une constante est réalisée par un appel à la fonction *define*
  - define("NOM_CONST",valeur)
    - Exemple:

    Define("HEURE_FIN","21h30");

    Echo "commandez avant ", HEURE_FIN;
  - Constantes prédéfines
    - TRUE, FALSE
    - Spéciales, pour l'affichage de messages d'erreurs, elles commencent et finissent par "__"
      - __FILE__ : donne le nom du fichier du script exécuté
      - __LINE__: fournit le num. de la ligne sur laquelle elle apparaît
      - __FUNCTION__ : donne le nom de la fonction dans laquelle la constante apparaît
      - PHP_VERSION : fournit la version de PHP
      - PHP_OS : le système d'exploitation sur lequel le script PHP est exécuté

PHP : les bases

- Constantes

```
…
            <TITLE>Constantes</TITLE>
…
<DIV>
            <?php
            // ne fonctionne pas
            echo "Ceci est le fichier : __FILE__"."<BR>";
            // ne fonctionne pas
            echo "Ceci est le fichier : {__FILE__}"."<BR>";
            // fonctionne, "." est l'opérateur binaire de concaténation
            echo "Ceci est le fichier  : ".__FILE__."<BR>";
            echo "Ceci est la ligne : ".__LINE__."<BR>";
            echo "Version PHP : ".PHP_VERSION."<BR>";
            echo "Version OS : ".PHP_OS;
            ?>
</DIV>
…
```

PHP : les bases

- Apostrophes et guillemets

En PHP, les valeurs entourées d'apostrophes sont traitées littéralement alors que celles entre guillemets sont interprétées.

Faire le test en déclarant une variable : $var = 'test';
Et en affichant :

```
echo "var est égal à $var";
echo 'var est egal à $var';
echo " \$var est égal à $var";
echo '\$var est égal à $var';
```

\ est la caractère d'échappement.

PHP : les bases

- Chaînes de caractères
  - Syntaxe

Les chaines s'écrivent entre guillemets, apostrophes ou avec la syntaxe here-doc <<< qui permet d'écrire une chaine de caractère sur plusieurs lignes.

Il est possible d'accéder à un caractère de la chaine en précisant son indice, exemple : $ch{8}

```
<?php
$nb = 3;
$ch1 = "$nb portions de frites offertes";
$ch2 = 'pour toute commande ($nb)';
$ch3 = <<<promo
<BR> Aujourd'hui offre exceptionnelle : <BR>
<EM>$nb portions de frites offertes</EM><BR>
pour toute commande
promo;

echo "$ch1 $ch2<BR>";
echo $ch3,"<BR>";
echo $ch2{3};
?>
```

Des séquences spéciales sont définies, elles sont précédées par le caractère "\"

Séquence d'échappement	Signification
\"	Caractère guillemet
\n	Saut de ligne (début d'une nouvelle ligne)
\r	Retour chariot (fin d'une ligne)
\t	Tabulation horizontale
\\	Caractère \

PHP : les bases

- Chaînes de caractères

Affichez le code Source HTML de ces deux scripts PHP :

```
<?php
for ($i=0;$i<5;$i++) {
   echo "test d'affichage<BR>";
}
?>
```

```
<?php
for ($i=0;$i<5;$i++) {
   echo "test d'affichage<BR>\n";
}
?>
```

Il existe un grand nombres de fonctions prédéfinies pour le traitement des caractères

Par exemple, testez la fonction *strrev() :*

```
echo strrev("Bonjour le monde!");
```

HTML : notions de base

Exercice 2:

Ouvrez nombre.php dans votre éditeur et réécrivez l'instruction echo() originale en utilisant des guillemets

PHP : les bases

- Apostrophes et guillemets

Comme le HTML valide inclut souvent une grande quantité d'attributs avec des guillemets, il est souvent plus facile d'utiliser des apostrophes lors de l'impression de HTML en PHP:

```
...
        echo '<TABLE width="80%" border="4" cellspacing="2" cellpading="3" align="center">';
        echo '<TR><TD> Une seule cellule </TD></TR>';
        echo '</TABLE>';

...
```

Exercice 3 :

Ecrivez ce code HTML en utilisant des guillemets

PHP : les bases

- Opérateurs arithmétiques
  - $x, $y, $z sont trois variables dont les valeurs sont : 8,5,-3

Opérateur	Expression	Evaluation
+	$x+$y	13
-	$x-$y	3
*	$x*$y	40
/	$x/$y	1.6
% (opérateur modulo)	$x%$y (Donne le reste de la division entre 2 nombres)	3

- Opérateurs relationnels

Opérateur	Expression	Evaluation
== (égalité)	2 == 2 2 == "2" 2 == 8	TRUE TRUE FALSE
=== (égalité et même type)	2 === 2 2 === "2" 2 === 8	TRUE FALSE FALSE
!= (différence)	2 != 8 2 != 2 2 != "2"	TRUE FALSE FALSE
!== (différence entre 2 types)	2 ! == 8 2 ! == "2" 2 ! == 2	TRUE TRUE FALSE
>=, >, <, <=		

PHP : les bases

- Opérateurs de concaténation
  - L'opérateur binaire de concaténation "." s'applique uniquement aux chaines de caractères.
- Opérateurs logiques
  - Prenons *expv* qui est TRUE et *expf* qui est FALSE

Opérateur	Expression	Evaluation
! (négation)	!expv !expf	FALSE TRUE
|| (or, disjonction)	expv || expf expv || expv expf || expv expf || exprf	TRUE TRUE TRUE FALSE
Xor (ou exclusif)	expv xor expf expv xor expv expf xor expv expf xor exprf	TRUE FALSE TRUE FALSE
&& (and, conjonction)	expv && expf expv && expv expf && expv expf && exprf	FALSE TRUE FALSE FALSE

PHP : les bases

- Opérateurs d'affectation et d'affectation élargie

Opérateur	Code	Equivalent
+= -= *= /= .=	Exemple : $x += exp	$x = $x + exp

- Opérateur d'incrémentation et de décrémentation

Opérateur	Code	Exemple
++ (incrémente de 1) -- (décrémente de 1)	$x = 3; Echo $x++, " "; Echo $x, " "; $x = 3; Echo ++$x, " "; Echo $x, " "; $x = 3; Echo $x--, " "; Echo $x, " "; $x = 3; Echo --$x, " "; Echo $x, " ";	 3 4 4 4 3 2 2 2

PHP : les bases

- Conditions
  - If … else

```
If (expr)
    {Instruction1}
Else
    {Instruction2}
```

Une condition peut-être vraie en PHP si elle possède une valeur autre que 0, une chaîne vide, FALSE ou NULL.

Exercice 4 :

Reprenons le script 'horaires de livraison', en utilisant une instruction conditionnelle, nous pouvons réaliser un compte à rebours qui informe le client du nombre d'heures et de minutes qu'il reste pour passer commande s'il a émis la requête pendant les heures de commandes (18h à 21h30). En dehors de ces heures, il obtient un message lui indiquant l'heure courante et les horaires de livraison.

Ecrire le script PHP traduisant l'algorithme proposé.

Algorithme :

Définir les constantes suivantes :
HEURE_DEB ← 18
MIN_DEB ← 0
HEURE_FIN ← 21
MIN_FIN ← 30

Stocker les variables heure et minutes courantes :
$h_courante ← Date("H ")
$min_courante ← Date("i ")
Afficher l'heure courante

Soit les variables suivante :

*$h_cou_min = $h_courante*60 + $min_courante désigne l'heure courante en minute*

*$h_deb_min = HEURE_DEB*60+MIN_DEB désigne l'heure de début en minute*

*$h_fin_min = HEURE_FIN*60+MIN_FIN désigne l'heure de fin en minute*

Conversion :
SI $h_cou_min > $h_deb_min ET $h_cou_min < $h_fin_min
$reste = $h_fin_min - $h_cou_min;
$r_h = intval($reste/60);
$r_m = $reste % 60;
Afficher le nombre d'heure (r_h) et de minutes (r_m) qu'il restent pour commander
Sinon
Afficher nous livrons de 18h à 22h les plats commandés avant 21h30";

Définir les constantes suivantes :

- *HEURE_DEB ← 18*
- *MIN_DEB ← 0*
- *HEURE_FIN ← 21*
- *MIN_FIN ← 30*
-
- *Stocker les variables heure et minutes courantes :*
- *$h_courante ← Date("H ")*
- *$min_courante ← Date("i ")*
- *Afficher l'heure courante*
-
- *Soit les variables suivante :*
- *$h_cou_min = $h_courante*60 + $min_courante désigne l'heure courante en minute*
- *$h_deb_min = HEURE_DEB*60+MIN_DEB désigne l'heure de début en minute*
- *$h_fin_min = HEURE_FIN*60+MIN_FIN désigne l'heure de fin en minute*
-
- *Conversion :*
- *SI $h_cou_min > $h_deb_min ET $h_cou_min < $h_fin_min*
- *$reste = $h_fin_min - $h_cou_min;*
- *$r_h = intval($reste/60); \\Retourne la valeur numérique entière*
- *$r_m = $reste % 60;*
- *Afficher le nombre d'heure (r_h) et de minutes (r_m) qu'il restent pour commander*
- *Sinon*
- *Afficher nous livrons de 18h à 22h les plats commandés avant 21h30";*

PHP : les bases

- Conditions

Exercice 5 :

Modifiez le script précédent pour obtenir ceci (temps restant en rouge) lorsqu'il reste moins de 30 minutes.

Algorithme :

Si $r_h=0 ET $r_m<30 alors <font color=red>

PHP : les bases

- Conditions

  - Opération conditionnel "?"

    exp1 ? exp2 : exp3 ➔ Si exp1 est TRUE, l'opérateur conditionnel évalue exp2 sinon exp3

    Exemple : $reste = $h_cou_min < $h_fin_min ? $h_fin_min - $h_cou_min :0;

    Exercice 6 :

    Modifier l'exercice précédent ('temps restant en rouge') en utilisant l'opérateur "?"

PHP : les bases

- Conditions
  - if … elseif … else

If (exp1)
    instruction1
Elseif (exp2)
    instruction2
…
Elseif (exprN)
    instructionN
Else
    instructionN+1

➔ cette instruction évalue exp1 si elle est TRUE, exécute instruction1. Sinon l'instruction dépendant du premier *elseif* qui produit TRUE est exécutée. Lorsque tous les tests ont produit la valeur FALSE, c'est l'instruction du *else* qui est exécutée.

```
…               <TITLE>Promotions</TITLE>
 …
<H1>Offre du jour</H1>
<DIV>
                <?php
                $jour = date("w");                  // w = un code de jour de 0 à 6
                if ($jour == 0) {  // dimanche
                                echo "dimanche : la bi&egrave;re est offerte avec la
choucroute.";
                }
                elseif ($jour == 1) {               //lundi
                                echo "lundi : 1 dessert offert pour toute commande.";
                }
                elseif ($jour == 2) {               //mardi
                                echo "mardi : 1 verre offert pour toute commande.";
                }
                else {                              //les autres jours
                                echo "deux portions de frites pour le prix d'une.";
                }
                ?>
</DIV> …
```

PHP : les bases

- Conditions
  - Fonction isset($var)

Cette fonction vérifie si une variable est définie, autrement dit si elle a une valeur autre que NULL (NULL est un type spéciale en PHP, qui représente l'absence de toutes valeur définie).

```php
<?php

$var = '';

// Ceci est vrai, alors le texte est affiché
if (isset($var)) {
    echo 'Cette variable existe, donc je peux l\'afficher.';
}

// Dans les exemples suivants, nous utilisons var_dump() pour afficher
// le retour de la fonction isset().

$a = 'test';
$b = ' un autre test';

var_dump(isset($a));      // TRUE
var_dump(isset($a, $b)); // TRUE

unset ($a);

var_dump(isset($a));     // FALSE
var_dump(isset($a, $b)); // FALSE

$foo = NULL;
var_dump(isset($foo));   // FALSE

?>
```

PHP : les bases

- Itérations
  - While …

while (exp)
instruction

➔ répète une instruction unique ou un bloc tant que l'expression évaluée exp est vraie.

```
$x = 2; // initialisation des variables
$i  = 0;
while ($i<5){
   $x *= $x;
   echo $x, "<BR>";
   $i++;
}
```

  - Do … While

Do
instruction
While(exp);

➔ répète une instruction unique ou un bloc tant que l'expression exp est vraie. La ≠ avec *while* est que le test est réalisé après l'exécution des instructions dans la boucles.

```
$x = 2;
$i = 0; // faire le test avec $i = 5
 do {
    $x *= $x;
    echo $x, "<BR>";
    $i++;
}
while ($i<5);
```

PHP : les bases

- Itérations
  - For …

    for (exp1; exp2; exp3) {
    instruction1;
    }

```
for ($i=0, $x=2;$i<3;$i++) {
   $x *= $x;
   echo $x, "<BR>";
}
```

    ➔ exécute l'instruction ou le bloc répétitivement.
    exp1 effectue des initialisations
    exp2 est le test de continuation
    exp3 est évaluée à la fin de chaque itération

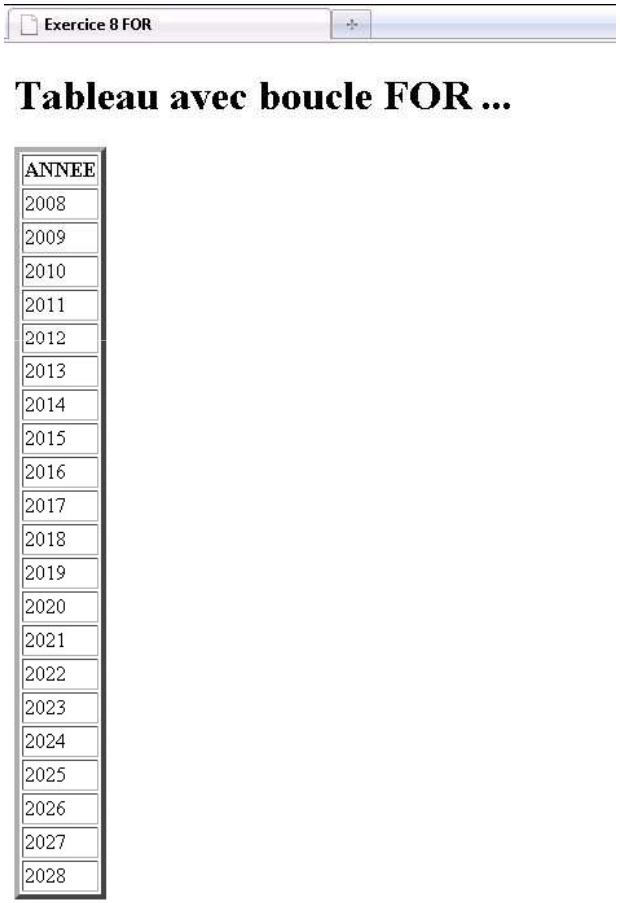
  - Foreach …

    Utilisée avec les tableaux, sera présentée plus loin …

PHP : les bases

Exercice 8 :

En utilisant une boucle For, remplir un tableau avec une colonne ANNEE allant de 2008 à 2028.



PHP : les bases

- Fonctions

PHP inclut un grand nombre de fonctions prédéfinies, mais il permet aussi de définir et d'utiliser vos propre fonctions.

Une fonction utilise aucun ou plusieurs arguments pour réaliser un traitement et retourne éventuellement un résultat.

- Déclaration d'une fonction

```
function nom_fonction(liste_parametres) {    // les noms ne tiennent pas compte de la casse en PHP
                    // code PHP de la fonction
}
```

L'instruction optionnelle *return;* ou *return exp;* dans le corps de la fonction provoque l'abandon de la fonction et le retour à l'instruction qui a appelé la fonction. Lorsqu'une expression exp est donnée, sa valeur est évaluée et est renvoyée à l'instruction qui a effectué l'appel.

- Appel d'une fonction

```
$res = nom_fonction(liste_parametres);
nom_fonction(liste_parametres);
```

```
<?php
// Déclaration de la fonction
function table($i_passe,$res_passe){
                    echo "$i_passe*3 --> <STRONG>$res_passe</STRONG><BR>";
}

echo "Table de trois<br><br>";
for ($i=1;$i<=10;$i++){
                    $res=$i*3;
                    // appel de la fonction table()
                    table($i,$res);
}
?>
```

Exercice 9 :

Modifiez l'exercice 'horaires de livraison' en définissant la fonction suivante qui retourne une heure convertie en minutes.

```
function convert_min($heure,$min){          // Paramètres passés par valeur, utilisation de return
                    return ($heure*60+$min);
                }
```

PHP : les bases

- Fonctions
  - Valeurs par défaut

```
<?php
// fonction
function tva($prix_HT,$taux = 19.6)                    {
   $prix_TTC = ($prix_HT*$taux)/100+$prix_HT;
   echo $prix_TTC," €<BR>";
}
// appels
echo "TVA 19,6% : ",tva(10);
echo "TVA 5,5% : ",tva(10,5.5);
?>
```

  - Portée des variables globales et locales
    - Les variables globales sont statiques, elles sont visibles dans tout le script, sauf dans les fonctions.

```
<?php
function test() {
   $h = 100;
   echo "Dans la fonction : ",$h,"<BR>";
}
$h = 0;
test();
echo "En dehors de la fonction : ",$h,"<BR>";
?>
```

    - Pour utiliser une variable globale dans une fonction il faut préciser dans la fonction que la variable est globale en utilisant le mot-clé *global.*

```
<?php
function test() {
   global $h;
   $h = 100;
   echo "Dans la fonction : ",$h,"<BR>";
}
$h = 0;
test();
echo "En dehors de la fonction : ",$h,"<BR>";
?>
```

PHP : les bases

- Tableaux
  - Un tableau est composé d'éléments qui ne sont pas forcément du même type.

    Chaque élément est stocké dans une case du tableau.

    Le tableau comporte des clés qui permettent d'accéder aux valeurs.

    Nous parlerons de tableaux avec accès par indice lorsque les clés sont des nombres entiers, et de tableaux associatifs lorsque les clé sont de chaines de caractères.

  - Accès par indice
    - A une dimension

      Pour accéder à une case, il suffit de donner le nom du tableau suivi du numéro de la case entre crochets, par exemple $prix[2] donnera le contenu de la case 2 du tableaux $prix ci-dessous.

Valeur	**45**	**154**	**58**	**78**	**31**	**5**	**74**
Index	**0**	**1**	**2**	**3**	**4**	**5**	**6**

Initialisation et ajout de cases

```
        $tab = array("b","o","n");
Ou      $tab[0] = "b"; $tab[1] = "o"; $tab[2] = "n";
Ou      $tab[] = "b"; $tab[] = "o"; $tab[] = "n";
```

PHP : les bases

- Tableaux

Parcours :

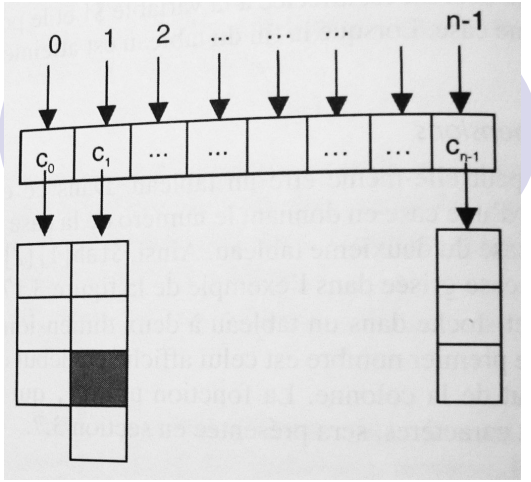
l'instruction foreach ($tab,$val) permet d'accéder à chaque case non nulle du tableau $tab et de stocker sa valeur pour une itération dans $val.

```
<?php
$tab[0] = "b"; $tab[1] = "o"; $tab[3] = "n"; // pas de case 2
foreach ($tab as $i) {
echo "$i-";
}
echo "<BR>";
$t = count($tab);  //renvoie le nombre d'élément du tableau
?>
```

L'instruction *foreach* est capable d'afficher tout le contenu du tableau, lorsque la fin du tableau est atteinte, les itérations cessent.

PHP : les bases

- Tableaux
  - À plusieurs dimensions



Une case d'un tableau $tab peut elle-même être un tableau.

Dans ce cas, il sera possible d'accéder au contenu d'une case en donnant le numéro de la case du tableau principal, suivi de la case du deuxième tableau. Par exemple $tab[1][2]

```
<?php
// calculer et stocker
   for ($i=0;$i<=10;$i++) {
      for ($j=0;$j<=10;$j++) {
         $add[$i][$j] = $i + $j;
      }
   }

// Afficher
echo "<BR>";
foreach ($add as $ligne) {
   foreach ($ligne as $col) {
      printf("%02d ", $col); //printf() Affiche une chaîne de caractères formatée

      }
      echo "\n<BR>";
}
?>
```

Tab_parcours

00 01 02 03 04 05 06 07 08 09 10
01 02 03 04 05 06 07 08 09 10 11
02 03 04 05 06 07 08 09 10 11 12
03 04 05 06 07 08 09 10 11 12 13
04 05 06 07 08 09 10 11 12 13 14
05 06 07 08 09 10 11 12 13 14 15
06 07 08 09 10 11 12 13 14 15 16
07 08 09 10 11 12 13 14 15 16 17
08 09 10 11 12 13 14 15 16 17 18
09 10 11 12 13 14 15 16 17 18 19
10 11 12 13 14 15 16 17 18 19 20

PHP : les bases

- Tableaux
  - Associatifs
    - Un tableau associatif associe une clé et une valeur, cette clé peut être un entier (ce qui revient aux tableaux avec accès par indice) ou une chaîne de caractères. Ainsi il sera possible de demander la valeur de la case "pizza" d'un tableau $tarif en écrivant $tarif["pizza"].

    - Initialisation

      $nomtab["cle"] = valeur;

      $nomtab = array("cle1"=>valeur1,"cle2"=>valeur2,…);

```
<?php
//initialisation 1
$tarif["pizza napolitaine"]=6;
$tarif["pizza royale"]=8;
$tarif["pizza aux fruits de mer"]=8.50;
$tarif["moule frites"]=7.30;
$tarif["poulet frites"]=6;

//initialisation 2
$stock = array("poulets frites"=>50, "pizzas"=>22, "moules frites"=>8);

//acces a une valeur
echo $tarif["poulet frites"]." € <BR>";
echo $stock["poulets frites"], " portions<BR>";
?>
```

PHP : les bases

- Tableaux
  - Opérateur d'ajout :

    L'opérateur "+" ajoute au tableau de gauche le tableau de droite, sans dupliquer les cases dont les clés sont identiques.

```php
<?php
//initialisation
$tarif = array("pizza"=>6, "pizza royale"=>8, "poulet"=>6);
$tarif2 = array("couscous"=>7, "pizza"=>6.5, "lasagnes"=>7.50);

$res = $tarif + $tarif2;

//parcourir le tableau avec foreach
foreach ($res as $cle => $val) {
   echo "<BR>$cle = $val portions ";
}
?>
```

Il existe un grand nombres de fonctions prédéfinies pour les tableaux.

Par exemple, testez les fonctions :

asort(array t) qui trie le tableau associatif t par ordre croissant de valeurs.

ksort(array t) qui trie le tableau associatif t par ordre croissant de clés.

PHP : les bases

- Tableaux

Exercice 10 :

Soit deux tableaux (accès par indice, une dimension), $moisFrancais et $cellColor qui contiennent respectivement les noms de mois d'une année (Janvier, Février, Mars, Avril, Mai, Juin, Juillet, Août, Septembre, Octobre, Novembre, Décembre) et des noms de couleurs (blue, white, red, yellow, grey, lime, lightblue, fuchsia, lightgrey, olive, pink, purple)

1) Construire ces deux tableaux en déclarant les variables $moisFrancais et $cellColor.
2) Afficher le contenu de ces deux tableaux en utilisant la boucle For …
3) Voici ce que l'on souhaite obtenir : (mémo <TD bgcolor=nom de la couleur>)

PHP : les bases

Exercice 10 (Pour vous mettre sur la piste) :

```
$moisFrancais = array('','Janvier','Février','Mars','Avril','Mai','Juin','Juillet','Aout','Septembre','Octobre','Novembre','Décembre');
$cellColor = array('','blue','white','red','yellow','grey','lime','lightblue','fuchsia', 'lightgrey','olive','pink','purple');

echo "<table border=1> ";
For(---;---;---)
{
    echo "<td bgcolor=---> --- </td><td>----</td>" ;
        ---
}
echo "</table>";
```

PHP : les bases

Création d'un formulaire - rappel

La gestion des formulaires HTML est un processus très important pour les site web dynamique.

Elle s'effectue en 2 étapes :

- La création du formulaire HTML
- La création du script PHP correspondant qui récupérera les donnée du formulaire

Rappel sur les formulaires HTML

Un formulaire HTML se srée en utilisant les balises *form* et différents éléments permettant de récupérer un entrée.

La balise *form ressemble à ceci :*

Formulaire HTML

```
…
<FORM action= " script.php" method="POST" name="nom_du_form">
  ….
</FORM>
…
```

L'attribut method d'un formulaire indique comment les données sont envoyées à la page qui les gère.
Deux méthodes (GET et POST)
GET : méthode qui envoie les données à la page qui doit les réceptionner sous la forme d'une série de paires ajoutée à l'URL.
Exemple : http://localhost/script.php?nom=Homer&genre=M&age=35
Avantage :
- peut-être placée dans les favoris
- utilisation Précédent possible ou recharger la page

Inconvénient :
- Quantité de données qu'il est possible de transférer est limitée
- Moins sécurisée (données visibles)

GET est le plus souvent utilisée pour demander des informations ou le résultat d'une recherche

POST : méthode utilisée lorsqu'une action est requise.
Par exemple mise à jour d'un enregistrement de BD, envoi d'un email

PHP : les bases

- Création d'un formulaire HTML - méthode

1. Commencez un nouveau document HTML.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<HTML>
<HEAD>
		<META name="author" content="Eddy Barile">
		<META http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
		<META http-equiv="Content-Type" content="text/html; charset=ISO-8859-15">
		 <TITLE> Simple Formulaire HTML </TITLE>
</HEAD>
<BODY>
<!-- Script form.html -->
```

2. Ajoutez la balise de formulaire initiale.

```
<FORM action="manip_form.php" method = "POST">
```

3. Commencez le formulaire

```
<FIELDSET><LEGEND>Entrez les informations dans le formulaire ci-dessous:</LEGEND>
```

PHP : les bases

- Création d'un formulaire HTML - méthode

4. Ajoutez deux entrées texte.

```
<P><B>Nom:</B> <INPUT type="text" name="nom" size="20" maxlength="40"></p>
<P><B>Email :</B> <INPUT type="text" name="email" size="40" maxlength="60"></p>
```

5. Ajoutez une paire de boutons radio.

```
<P><B>Genre:</B> <INPUT type="radio" name="genre" value="M"/>Masculin <INPUT type="radio" name="genre" value="F"/>Féminin
```

6. Ajoutez un menu déroulant.

```
<P><B>Age:</B>
<SELECT name="age">
                <option value="0-29">Moins de 30</option>
                <option value="30-60">Entre 30 et 60</option>
                <option value="60+">Plus de 60</option>
</SELECT></P>
```

PHP : les bases

- Création d'un formulaire HTML - méthode

7. Ajoutez une zone de texte pour les commentaires.

```
<P><B>Commentaires:</B> <TEXTAREA name="commentaires" rows="3" cols="40"></TEXTAREA></P>
```

8. Terminez le formulaire

```
<FIELDSET>
<DIV align="center"><INPUT type="submit" name="submit" value="Valider les informations"></div>
</FORM>
```

9. Terminez la page HTML

```
</BODY>
</HTML>
```

10. Enregistrez le fichier sous le nom form.html, placez-le dans votre répertoire web et observez-le dans votre navigateur web.

PHP : les bases

- Création d'un formulaire Script PHP - méthode
  1. Créez un nouveau document PHP

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<HTML>
<HEAD>
                <META name="author" content="Eddy Barile">
                <META http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
                <META http-equiv="Content-Type" content="text/html; charset=ISO-8859-15">
                 <TITLE> Formulaire PHP Feedback </TITLE>
</HEAD>
<BODY>
<!-- Script form.html -->
```

  2. Ajoutez la balise PHP ouvrante et créez une version raccourcie des variables de données de formulaire

```
<?PHP #script manip_form.php
$nom = $_REQUEST['nom'];
$email = $_REQUEST['email'];
$commentaires = $REQUEST['commentaires'];
```

$_REQUEST est une variable qui stocke toutes les données envoyées à une page PHP via la méthode GET ou POST. L'orthographe et la casse de vos nom de variable doivent ici correspondre exactement aux valeurs des noms dans le formulaire HTML

PHP : les bases

- Création d'un formulaire Script PHP - méthode

3. Imprimez les valeurs de nom, d'email et de commentaires reçues

```
Echo "<P>Merci, <b>$nom </b>, pour ce commentaire : <br><tt>$commentaires</tt></P>
<P>Nous vous répondrons sur cet email : <i>$email</i>.</P>\n";
```

4. Complétez le code HTML et PHP

```
?>
</BODY>
</HTML>
```

5. Enregistrez le fichier sous le nom manip_form.php et placez-le dans le même répertoire web que form.html

6. Testez les deux documents en chargeant form.html, en remplissant et envoyant le formulaire.

PHP : les bases

- Création d'un formulaire HTML+Script PHP (tout en un) – méthode

```
<?php #script manip_form.php
if (ISSET($_REQUEST['nom'])) {          // test de l'existence de la variable
        $nom = $_REQUEST['nom'];
        $email = $_REQUEST['email'];
        $commentaires = $_REQUEST['commentaires'];
        Echo "<P>Merci, <b>$nom </b>, pour ce commentaire : <br><tt>$commentaires</tt>
        <P>Nous vous répondrons sur cet email : <i>$email</i>.</p>\n";
}
Else { // la variable n'existe pas, nous affichons le formulaire
?>
<FORM action=\"manip_form2.php\" method = "POST">
<FIELDSET><LEGEND>Entrez les informations dans le formulaire ci-dessous:</LEGEND>
<P><B>Nom:</B> <INPUT type="text" name="nom" size="20" maxlength="40"></p>
<P><B>Email :</B> <INPUT type="text" name="email" size="40" maxlength="60"></p>
<P><B>Genre:</B> <INPUT type="radio" name="genre" value="M"/>Masculin <INPUT type="radio"
name="genre" value="F"/>Féminin
<P><B>Age:</B>
<SELECT name="age">
        <option value="0-29">Moins de 30</option>
        <option value="30-60">Entre 30 et 60</option>
        <option value="60+">Plus de 60</option>
</SELECT></P>
<P><B>Commentaires:</B> <TEXTAREA name="commentaires" rows="3"
cols="40"></TEXTAREA></P>
<FIELDSET>
<DIV align="center"><INPUT type="submit" name="submit" value="Valider les informations"></div>
</FORM>
<?php }
?>
```

PHP : les bases

Conclusion

- Nous avons vu :
  - Envoi de chaînes de caractères vers le navigateur avec *echo* et *print*
  - Chaîne de caractères
  - Variables et constantes
  - Opérateurs (relationnels, logiques, …)
  - Structures de contrôle (itération, condition, …)
  - (Fonction et durée de vie des variables globales, locales et statique)
  - Tableaux
  - Accès aux données envoyées par le client depuis un formulaire
  - Fusionner dans un seul fichier l'affichage et le traitement du formulaire