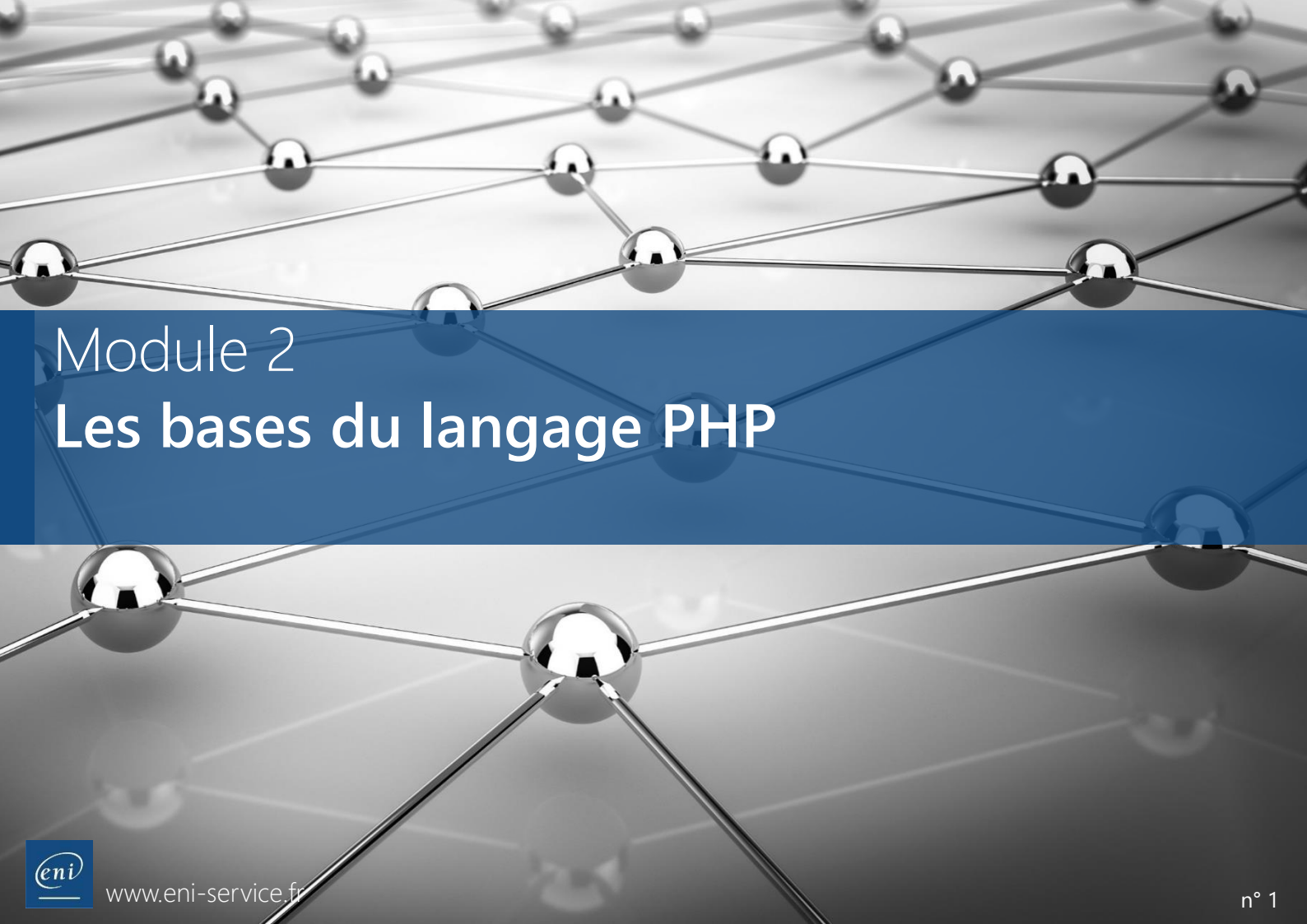


Module 2

Les bases du langage PHP

Contenu du module

- Intégration de PHP avec les pages HTML.
  - Approches pour mixer PHP et HTML.
  - Bonnes pratiques de structuration d'applications.
- La syntaxe de base du langage.
  - Les types de données.
  - Les variables et les tableaux.
  - Les constantes.
  - Les opérateurs.
  - Le "spaceship operator" de PHP 7 pour les comparaisons combinées.
  - Les structures de contrôle

Intégration de PHP avec les pages HTML

- Le code PHP est inclus dans une page HTML à l'intérieur de balises
  - Appelées également *tags*
- Deux syntaxes :
  - `<?php ... ?>`
  - `<? ... ?>`
- La première syntaxe est celle recommandée
- La deuxième syntaxe ne peut être utilisée que si la directive `short_open_tag` est positionnée à on dans le fichier `php.ini`
  - Elle est déconseillée et simplement conservée pour compatibilité avec d'anciennes applications
- Si le script ne contient que du code PHP, la balise de fermeture peut être omise
  - C'est une approche très courante !

Approche pour mixer PHP et HTML

- Plusieurs approches pour mixer du PHP et du HTML
- Deux principes très simples :
  - La page peut contenir une ou plusieurs inclusions de code PHP
  - Le code PHP génère du "texte" qui est intégré dans la page HTML envoyée au navigateur.
    - Tout "texte" compréhensible par le navigateur peut donc être généré par le code PHP : du texte simple, du code HTML, du code JavaScript...
- Approche courante :
  - Utiliser PHP uniquement pour générer la partie réellement dynamique de la page
  - Le reste est directement écrit en HTML dans le fichier.
- Cette technique rend le code moins lourd et permet de voir tout de suite où se trouve la logique applicative.
- Important :
  - Le document HTML envoyé au navigateur doit être valide et si possible conforme aux standards HTML5 !

Exemple de page complète

Déclarations de variables utilisées dans d'autres blocs…

Une structure peut commencer dans un bloc…

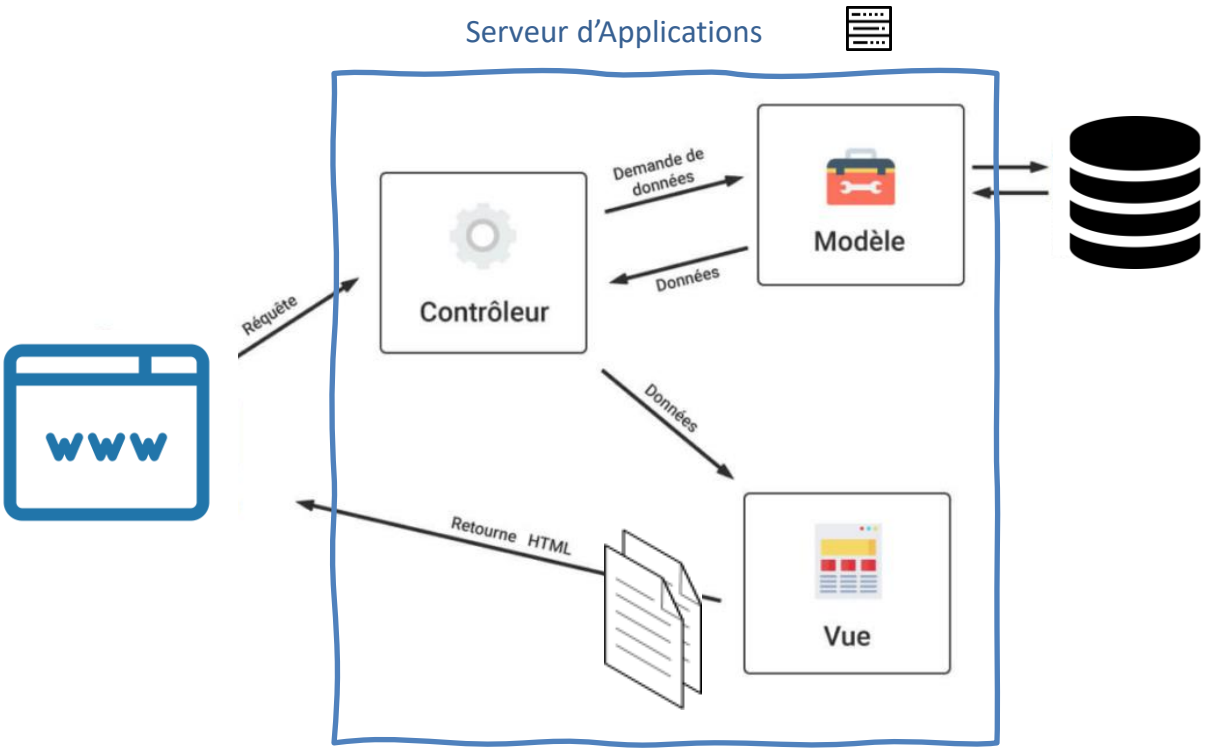
… et se terminer dans un autre

```php
<?php
    $aujourdhui = date("d/m/Y");     // Date du jour
    $heure = date("H:i:s");          // Heure du jour
    $heure_seule = date("H");        // Heure seule
?>
<<!DOCTYPE html>
<html lang="fr">
    <head>
        <title>PHP est vraiment cool !</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1">
    </head>
    <body>
        <h1>PHP est vraiment cool !</h1>
        <p>
            <?php
                if($heure_seule >= 17) {
            ?>
                    Bonsoir,
            <?php
                } else {
            ?>
                    Bonjour,
            <?php
                }
            ?>
            nous sommes le <?php echo $aujourdhui; ?>, et il est <?php echo $heure; ?>
    </body>
</html>
```

Bonnes pratiques de structuration d'applications

- Historiquement, PHP devait permettre de développer rapidement
  - La maintenabilité et l'évolutivité étaient secondaires…
- Aujourd'hui cela est essentiel !
  - Séparation des responsabilités applicatives
- Plusieurs approches :
  - Modèle MVC
    - Pour les applications Web traditionnelles
  - Micro-services
- Mais des concepts similaires :
  - On ne mélange pas :
    - Logique de présentation et IHM
    - Traitements métiers
    - Accès aux ressources et aux données

Exemple d'architecture applicative : Le modèle MVC



Le langage PHP

Basics…

- echo
  - Permet d'afficher des données PHP dans le flux de la page HTML renvoyée
  - Syntaxe :
    - `echo(chaîne texte[, …])`
    - `echo chaîne texte[,...]`

- Commentaires
  - `// Commentaire sur une ligne`
  - `# Commentaire sur une ligne`
  - ```
    /* Commentaire
       sur plusieurs
       lignes
    */
    ```
    ```

Instructions et blocs

- ;
  - Obligatoire à la fin de chaque instruction
  - Erreur fatale en cas d'oubli
  - Plusieurs instructions peuvent être écrites sur la même ligne
    - Même si cela nuit généralement à la lisibilité…
- Blocs
  - Introduits par des { }
  - Marques une limite logique
    - Classes, structures de contrôle, fonctions, …
    - Imbrication possible

Règles et conventions de nommage

- Applicables pour tous les identificateurs PHP (variable, constante, fonction...)
- Commencer par une lettre ou un souligné (_)
  - Les variables commencent forcément par le caractère $
- Ensuite des lettres, des chiffres ou le caractère souligné.
  - Dans cette définition, une lettre représente toute lettre minuscule ou majuscule comprise entre a et z (a à z et A à Z)
  - Les caractères accentués sont autorisés (en théorie...)
  - Pas les caractères du type #$%& qui ont une signification spéciale dans le langage PHP
- Ne pas comporter d'espaces ni de sauts de lignes

Conventions de nommage

- Écriture des variables, fonctions, constantes et classes
  - Les noms des variables, des fonctions, des méthodes et des attributs sont écrits en « camelCase »
    - `$maVariable = 10;`
    - `function calculerValeurMoyenne($max) { ... }`
  - Les noms des constantes sont écrits en MAJUSCULE
    - `define('PI', 3.14);`
  - Les noms des classes sont écrits en « PascalCase », elles sont stockées dans un fichier qui porte le même nom
    - `class Utilisateur { ... }`
- Ecriture des tableaux
  - Les clés de tableaux associatifs sont écrits en « snake_case »
    - `$host = [ 'ip_addr' => '1.2.3.4' ];`

Les types de données

- Types scalaires : ne contenant qu'une seule valeur
  - Nombre entier
  - Nombre à virgule flottante
  - Chaîne de caractères
  - Booléen
- Types composés : pouvant contenir plusieurs valeurs (*ils seront détaillés plus loin*)
  - Tableau
  - Objet
- Types spéciaux :
  - NULL
  - Ressource
  - Fonction de rappel
  - Itérable

Les types scalaires

- Entier (`int`)
  - Signé sur 32 bits ou 64 bits
  - Le caractère _ peut être utilisé comme séparateur pour améliorer la lisibilité
- Nombre réels (`float`)
  - Notation décimale (*Le séparateur est le point*) : `1.345`
  - Notation scientifique (*x.yEz*) : `2.45E2`
- Chaine de caractères (`string`)
  - Suite de caractère encadrée par des " ou des '
  - Caractère d'échappement /
    - Rend sa signification au caractère situé juste après
- Booléen (`bool`)
  - Deux valeurs possibles : `TRUE` (ou `true`) et `FALSE` (ou `false`)

Les types spéciaux

- NULL (`null`)
  - Correspond au type d'une variable utilisée sans jamais avoir été initialisée
  - Possède la valeur `NULL` (ou `null`)
- Ressource (`resource`)
  - Correspond à une référence vers une ressource externe : fichier ouvert, connexion réseau, ...
  - Doit être fermé après usage
- Fonction de rappel (`callable`)
  - Des fonctions qui peuvent être passées en paramètre à d'autres fonctions
- Itérable (`iterable`)
  - Désigne une structure qui contient un ensemble de valeurs qui peuvent être parcourues avec `foreach`
  - Tableau ou objet implémente l'interface `Traversable`

Déclaration de type

- PHP permet de déclarer des types de données dans certains cas
  - Paramètres des fonctions et des méthodes, valeur de retour des fonctions et des méthodes, ou propriétés (attributs) des classes

`int`	La valeur doit être un entier
`float`	La valeur doit être un nombre à virgule flottante
`string`	La valeur doit être une chaîne de caractères
`bool`	La valeur doit être un booléen
`array`	La valeur doit être un tableau
`callable`	La valeur doit être le nom d'une fonction de rappel
`iterable`	La valeur doit être du type itérable (tableau ou classe qui implémente l'interface `Traversable`)
`object`	La valeur doit être un objet
`self`	La valeur doit être un objet (une instance) de la même classe que celle dans laquelle la déclaration de type est présente. Utilisable uniquement dans une classe
`parent`	La valeur doit être un objet (une instance) du parent de la classe dans laquelle la déclaration de type est présente. Utilisable uniquement dans une classe
`mixed`	La valeur peut être de n'importe quel type (y compris `NULL`). Apparu en **version 8**

Les variables

- Identifiées par le préfixe $
  - Suivent les règles et conventions évoquées précédemment
  - Sensibles à la casse
    - `$nom` est différent de `$Nom`
- Définies lors de leur première utilisation
  - Pas d'instruction spécifique pour créer une variable
- Typage dynamique
  - A l'affectation d'une valeur
  - Le type peut donc changer
- Portée
  - Le script dans lequel elle est définie
- Durée de vie
  - Durée d'exécution du script
    - Nettoyage mémoire automatique

Les constantes

- Zone mémoire identifiée par un nom qui contient une valeur lisible mais non modifiable dans le programme
  - Type scalaires et tableaux
- Deux syntaxes :
  - Fonction `define()`
    - `define('MA_CONSTANTE', 12);`
  - Mot clé `const`
    - `const MA_CONSTANTE = 12;`
- Portée
  - Le script dans lequel elle est définie
- Fonction associée
  - `defined()` : Permet de savoir si une constante est définie ou non
    - Prend en paramètre le nom de la constante sous forme d'une chaine de caractères et renvoi un booléen

Les opérateurs

- Un opérateur est un caractère ou une suite de caractères à laquelle la grammaire de PHP donne une signification particulière
- Les opérateurs PHP sont regroupés dans les familles suivantes :
  - Opérateurs arithmétiques
  - Opérateur de chaine de caractères
  - Opérateurs logiques
  - Opérateurs d'assignation
  - Opérateurs de comparaison
  - Opérateurs spécifiques

Opérateurs : Arithmétiques

- Opérateurs arithmétiques
  - Les opérateurs arithmétiques sont utilisés avec des valeurs numériques pour effectuer des opérations mathématiques courantes :

Opérateur	Nom	Exemple
+	Addition	$x + $y
-	Soustraction	$x – $y
*	Multiplication	$x * $y
/	Division	$x / $y
%	Modulo	$x % $y
**	Exponentiel	$x ** $y
-	Opposé	-$x
++	Incrémentation (pré ou post)	++$x ou $x++
--	Décrémentation (pré ou post)	--$x ou $x--

Opérateurs : Chaine de caractères

- Le seul opérateur de chaîne est l'opérateur de concaténation, le point (.)
- Syntaxe :
  - `$chaineComplete = $chaine1 . $chaine2;`
- Cette syntaxe retourne une chaîne égale à la première chaîne immédiatement suivie de la deuxième
  - Aucun séparateur n'est placé entre les deux chaînes
  - Des espaces de part et d'autres du point peuvent améliorer la lisibilité

Opérateurs : Logiques

- Opérateurs logiques
  - Les opérateurs logiques sont utilisés pour combiner les instructions conditionnelles :

Opérateur	Description	Exemple
`and ou &&`	Retourne Vrai si les deux énoncés sont vrais	`$x > 5 && $x < 10`
`or ou ||`	Retourne Vrai si l'un des énoncés est vrai	`$x > 5 || $x < 4`
`!`	Inverse le résultat, retourne Faux si le résultat est vrai	`!$result`

Opérateurs : Assignation

- Les opérateurs d'assignation sont utilisés pour assigner des valeurs aux variables :

Opérateur	Exemple	Equivaut à
=	$x = 5	
+=	$x += 3	$x = $x + 3
-=	$x -= 3	$x = $x - 3
*=	$x *= 3	$x = $x * 3
/=	$x /= 3	$x = $x / 3
%=	$x %= 3	$x = $x % 3
**=	$x **= 3	$x = $x ** 3
.=	$x .= "chaine"	$x = $x . 3

Opérateurs : Comparaison

- Opérateurs de comparaison
  - Les opérateurs de comparaison sont utilisés pour comparer deux valeurs :

Opérateur	Nom	Exemple
==	Égalité (valeurs identiques)	$x == $y
===	Égalité stricte (valeurs ET types identiques)	$x === $y
!=	Différence (valeurs différentes)	$x != $y
!==	Différence stricte (valeurs OU types différents)	$x !== $y
>	Supérieur à	$x > $y
<	Inférieur à	$x < $y
>=	Supérieur ou égal à	$x >= $y
<=	Inférieur ou égal à	$x <= $y

Opérateur de coalescence nulle – PHP 7

- Coalescence nulle (*null coalescing operator*)
  - Retourne le premier opérande non nul de la liste
  - Syntaxe :
    - `expression1 ?? expression2`
  - Retourne la valeur de `expression1` si `expression1` est non NULL ou la valeur de `expression2` dans le cas contraire.
    - Plus généralement, cet opérateur fonctionne avec une liste variable d'opérandes (`expression1 ?? expression2 ?? expression3 ...`) et il retourne le premier opérande non NULL de la liste.
- Affectation de coalescence nulle
  - Opérateur combiné
  - Syntaxe :
    - `$variable ??= expression2`
  - Equivalent à :
    - `$variable = $variable ?? expression2`

Opérateur de comparaison combinée

- Retourne un entier négatif, positif ou nul selon le résultat de la comparaison des deux opérandes
- Syntaxe :
  - `expression1 <=> expression2`
- Retourne
  - -1 :
    - Si `expression1` est strictement inférieur à `expression2`
  - 0
    - Si `expression1` est égal à `expression2`
  - +1
    - Si `expression1` est strictement supérieur à `expression2`

Opérateur Nullsafe – PHP 8

- L'opérateur nullsafe `$->` permet d'éviter l'invocation d'un membre si la référence d'objet est nulle
  - Le second opérateur n'est évalué que si le premier opérateur n'est pas nul
- Exemple :
  - `$foo = $a?->b();`
  - Si `$a` est nul, la méthode `b()` n'est pas appelée et `$foo` est mis à null

Les structures de contrôle

- Les structures de contrôle permettent d'intervenir sur le déroulement d'une séquence d'instructions.
- L'exécution, habituellement séquentielle, pourra donc se trouver modifiée.
- Il existe essentiellement 2 types de structures de contrôle :
  - Les structures conditionnelles
    - Elle permettent de conditionner l'exécution d'une série d'instruction
  - Les structures itératives
    - Plus communément appelées « boucles », elles permettent d'itérer sur un ensemble de valeurs, ou pour un certain nombre de fois.

La structure `if`

- Exécution conditionnelle d'instruction
  - L'expression de la condition est écrit entre parenthèses et renvoi un booléen
- Il est possible d'utiliser d'autre conditions à la suite de la première (elseif) et d'utiliser l'alternative

Syntaxe 1

```
if(condition_1) {
  instructions_1;
}
elseif(condition_2) {
  instructions_2;
}
else {
  instructions_n;
}
```

Syntaxe 2

```
if(condition_1):
  instructions_1;
elseif(condition_2):
  instructions_2;
else:
  instructions_n;
endif;
```

La structure `switch`

- Equivalente à des `if` – `elseif` multiples
  - Attention au `break` ! Ou bien les « case » suivants seront évalués

Syntaxe 1

```
switch(expression) {
  case expression_1: instructions_1;
                     [break;]
  case expression_2: instructions_2;
                     [break;]
  ...
  default: instructions_n;
}
```

Syntaxe 2

```
switch(expression):
  case expression_1: instructions_1;
                     [break;]
  case expression_2: instructions_2;
                     [break;]
  ...
  default: instructions_n;
endswitch;
```

La structure `while`

- Permet d'exécuter en boucle une série d'instructions tant qu'une condition est vraie

Syntaxe 1

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

Syntaxe 2

```
while(condition):
  instructions;
endwhile;
```

La structure do … while

- Permet d'exécuter en boucle une série d'instructions tant qu'une condition est vraie
  - Contrairement à la structure `while`, la condition est évaluée en sortie

- Une seule syntaxe est disponible

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

La structure `for`

- Comme dans le langage C, permet d'exécuter des instructions de manière itérative, en contrôlant les itérations à l'aide de trois expressions
  - `expression1` : Exécutée une fois au démarrage de la boucle.
  - `expression2` : Exécutée et évaluée comme booléen avant chaque itération
    - Si le résultat vaut TRUE, les instructions instructions sont exécutées
    - Si le résultat vaut FALSE, la boucle s'arrête
  - `expression3` : Exécutée à la fin de chaque itération.

Syntaxe 1

```
for(expression1; expression2; expression 3) {
  instructions;
}
```

Syntaxe 2

```
for(expression1; expression2; expression 3):
  instructions;
endfor;
```

`break` et `continue`

- `continue`
  - Peut être utilisée dans toutes les structures de contrôle itératives pour interrompre l'itération en cours et passer à l'itération suivante

- `break`
  - Peut être utilisée dans toutes les structures de contrôle itératives pour interrompre la boucle en cours
  - Peut aussi être utilisée dans une structure de contrôle `switch`

L'expression `match` – PHP 8

- Ressemble à un switch, mais c'est une expression !
  - Renvoie une valeur !
- Syntaxe :

```
$resultat = match(sujet) {
  comparaison[, ...] => valeur,
  [...]
  [default => valeur]
}
```

- Exemple :

```php
$return_value = match ($food) {
    'apple' => 'This food is an apple',
    'bar' => 'This food is a bar',
    'cake' => 'This food is a cake',
};
```

Les tableaux

- Une collection/liste d'éléments ordonnée de couples clé/valeur
- Un tableau peut être :
  - Numérique
    - Les clés sont des valeurs numériques, ordonnées et consécutives à partir de 0, elle est désignée par le terme indice
  - Associatif
    - Les clés ne sont pas forcément consécutives, ni ordonnées, et ce tableau peut présenter des clés entières et des clés de type chaîne
- La valeur associée à la clé peut être de n'importe quel type, y compris de type tableau ; dans ce cas, le tableau est dit multidimensionnel

Les tableaux : création

- Une variable de type tableau peut être créée en utilisant la fonction `array()` ou bien la notation à crochets (`[]`)
- Tableaux numériques :
  - `$tab = array('valeur1', 'valeur2', 'valeur3');`
  - `$tab = ['valeur1', 'valeur2', 'valeur3'];`
- Tableaux associatifs :
  - `$tab = array(10 => 'valeur1', 20 => 'valeur2', 30 => 'valeur3');`
  - `$tab = [10 => 'valeur1', 20 => 'valeur2', 30 => 'valeur3'];`
- Déclaration implicite
  - Une variable utilisée pour la première fois avec une notation de la forme `$variable[...]` est automatiquement créée avec le type tableau
    - `$tableau[] = valeur`
      - PHP recherche le plus grand indice entier utilisé et associe la valeur à l'indice immédiatement supérieur. Si le tableau est vide l'élément est placé à l'indice 0.
    - `$tableau[clé] = valeur`
      - PHP associe la valeur à la clé indiquée (qui peut être de type entier ou de type chaîne).

Les tableaux : accès

- La notation à crochets est utilisée pour accéder, en lecture ou en écriture, à un élément individuel du tableau
  - `$tableau[clé]`
- Pour les tableaux multidimensionnels, plusieurs séries de crochets doivent être utilisées
  - `$tableau[clé][clé]...`

Les tableaux : parcours

- PHP dispose d'une structure itérative spécialement conçue pour permettre le parcours d'un tableau (ou d'un itérable)
  - `foreach`
- Syntaxe :
  - On ne s'intéresse qu'aux valeurs :

```
foreach(tableau as variable_valeur) {
    instructions;
}
```

  - On s'intéresse aux clés et aux valeurs :

```
foreach(tableau as variable_clé => variable_valeur) {
    instructions;
}
```

Travaux pratiques