le cnam

PHP
Les bases

J-F Berger
J-F Dazy
P. Graffion

Décembre 2014

URL utiles

- Php : http://fr.php.net/docs.php

- Mysql : http://www-fr.mysql.com/
- Postgresql : http://www.postgresqlfr.org/

- Outils
  - WampServeur 2 :http://www.wampserver.com
  - Phpmyadmin : http://www.phpmyadmin.net/home_page/index.php

CGI : Common gateway interface *interface de passerelle commune*

- Un script **CGI** est un programme exécuté par le serveur web. Ce programme doit être dans un répertoire spécial (configuration du serveur)
- Pour traiter les formulaires HTML, l'utilisateur va choisir ou saisir des données, puis envoyer les données du formulaire en paramètre du script CGI.
- Deux méthodes : GET ou POST (deux requêtes http possibles)

```
<form action='http://nom_du_serveur/cgi-bin/script.cgi'  method=post ou get>
<input type=text name=nom>
<input type=text name=voisins>
<input type=submit value=envoi>
</form>
```

l'appui sur le bouton envoi génère la requête HTTP:

POST http://nom_du_serveur/cgi-bin/script.cgi HTTP/1.0
Content-Type: application/x-www-form-urlencoded
Content-Length: 21
Ligne blanche
nom=Dazy&voisins=aaaa : ensemble des couples
nom=valeur séparés par &

GET http://nom_du_serveur/cgi-bin/script.cgi?nom=Berger&voisins=bbbbb HTTP/1.0

L'info est la même, mais envoyée et récupérée différemment

CGI suite

- GET ou POST?
  - La méthode POST: pas de limite en taille (conseillée).
  - La méthode GET est limitée en nombre de paramètres : la taille URL+paramètres doit être inférieure à 1Kb.
  - La récupération des paramètres est faite
    - sur stdin pour la méthode POST (CONTENT_LENGTH donne la taille)
    - dans la variable d'environnement QUERY_STRING pour la méthode GET
  - Le script CGI doit décoder les données, les traiter, et fournir un résultat (html) au serveur qui l'envoie au client.
- Encodage :
  - Les caractères non ASCII (ceux dont le code est supérieur à 128) sont remplacés par la chaîne de caractères **%xx** où **xx** représente le code ASCII du caractère au format hexadécimal.
  - les caractères réservés sont également remplacés par leur valeur hexadécimale.
  - le caractère espace est remplacé par le caractère **+**.

CGI : Que doit faire le script? (1)

- **extraire l'information envoyée par le navigateur à l'aide du formulaire. La méthode dépend en fait de celle choisie dans le formulaire.**

```
#!/usr/bin/perl
# les données sont envoyées par méthode GET
# donc on récupère les données dans la variable d'environnement QUERY_STRING
$buffer=$ENV{"QUERY_STRING"};
# on split la chaine de données en des paires name=value
local(@champs) = split(/&/, $buffer);
# affichage du debut du code HTML (heredoc syntax)
$htmlStr = <<EOT;
Content-type: text/html\n\n
<html><head>
<title>Réponse au questionnaire</title>
</head>
<body BGCOLOR="#ffffff">
<H1>Résultat du traitement de votre questionnaire</H1>
<H2>Chaîne de données reçue par le programme</H2>
QUERY_STRING <STRONG>$buffer</STRONG>
<H2>Liste des informations décodées</H2>
<UL>
EOT
print STDOUT $htmlStr;
```

CGI : que doit faire le script? (2)

- **Décoder les données par couples**

```
print STDOUT "<UL>";
# récupération et mise en forme des données
# on parcourt la liste des paires name=value
foreach $i (0 .. $#champs)
  {  # On convertit les plus en espaces
    $champs[$i] =~ s/\+/ /g;
    # On sépare chaque champ en une clé et sa valeur
    ($key, $val) = split(/=/,$champs[$i],2);
     # On convertit les %XX de leur valeur hexadécimale en alphanumérique
    $key =~ s/%(..)/pack("c",hex($1))/ge;
    $val =~ s/%(..)/pack("c",hex($1))/ge;
     # on affiche le résultat  (ou on traite…)
      printf STDOUT "<LI><STRONG>%s:</STRONG>%s<LI>\n",$key,$val;
    }
print  "</UL></BODY></HTML>";
```

Avec la méthode POST :

read(STDIN,$buffer,$ENV{"CONTENT_LENGTH"});.

La variable CONTENT_LENGTH contenant la longueur de la chaîne constituée des couples nom=valeur.

PHP Historique

- Rasmus Lerdorf en 1994: crée une bibliothèque de scripts perl , puis C pour son usage personnel .
  - Son objectif:
    - Communiquer avec des BD
    - Générer des applications dynamiques pour le web
- Il publie son code sous Licence GNU Juin 95 : Personnal HomePage Tools
  - Améliore, développe d'autres outils PHP/FI (Forms Interpreter) 1997 Version béta
    - Variables de type perl,
    - Interprétation automatique des variables de formulaire,syntaxe intégrable dans html
- Réécriture complète (Andi Gutmans et Zeev Zuraski Juin 1998)
  d' où PHP3.0 Hypertext Preprocessor
- PHP 4.0 en Mai 2000 : Réécriture du moteur interne PHP (Zend Engine)
- PHP 5.0 en Juillet 2004 : nouveau modèle objet
- PHP 5.3 en Juin 2009 : espace de noms
- PHP 6 abandonné! (devait implémenter Unicode)

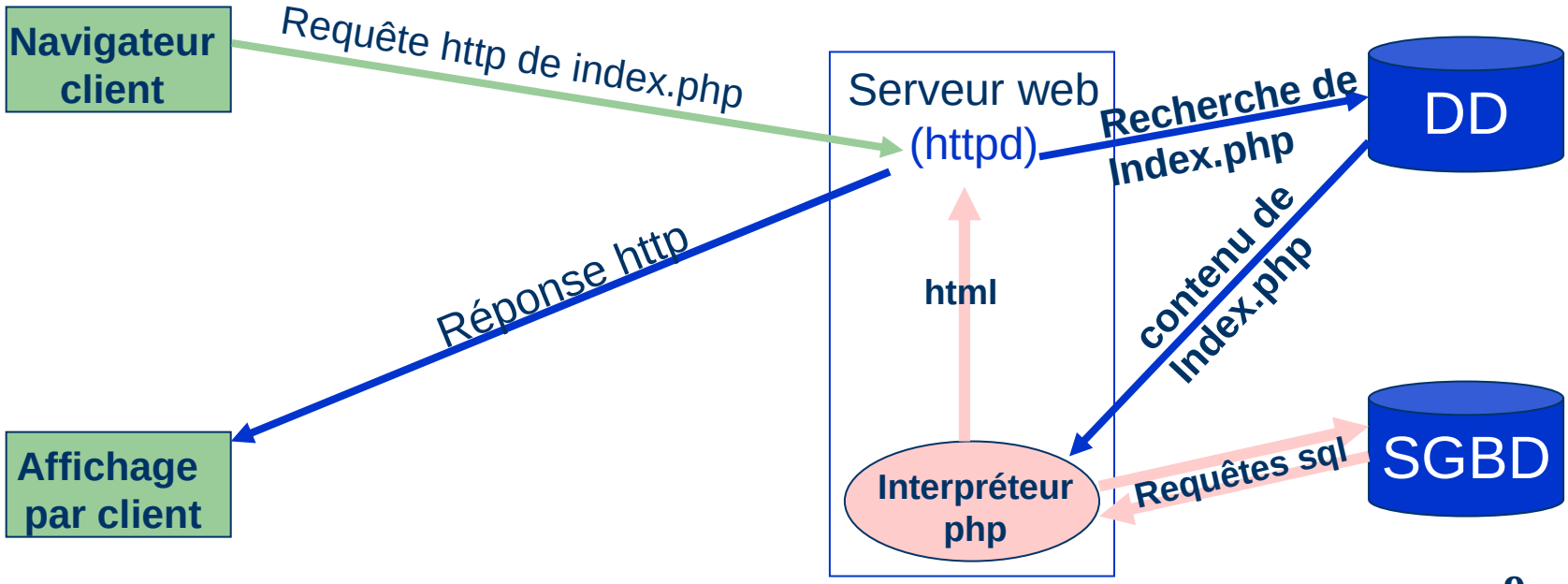
PHP en chiffres

- 4 500 000 développeurs
- 1 site Web sur 3 dans le monde
- 45 % des sites français
- 87% des entreprises du CAC 40

PHP : introduction

- Langage
  - Interprété indépendant de la plateforme d'exécution,
  - s'exécutant sur le serveur,
  - intégré au code html de la page
  - générant du html dynamique

Server-side, HTML embedded,cross-platform scripting language



PHP: principes et fonctionnement

- Le serveur lit les instructions php intégrées au code html
- Il les interprète, et les remplace par le résultat de leur exécution

AVANTAGES:

- Le client ne voit pas le source mais seulement le résultat(cf javascript…)
- Le code n'est pas lourd comme le cgi…
- La qualité dynamique est masquée.
- Indépendance du client

FONCTIONNEMENT

- Un bloc php est un groupe de lignes encadré par < ?php et ?>

(on peut se limiter à < ? et ?> mais ,….il faut alors avoir positionné short_open_tag dans la configuration php.ini)

- toute ligne à l'extérieur de ce couple n'est pas interprétée mais recopiée sur la sortie standard. Toute ligne à l'intérieur est interprétée comme une instruction php.

Ainsi les instructions php sont **invisibles en sortie** ;

- En cas d'erreur, un message est intégré au flux de sortie (interprétation interrompue sauf warning)

PHP et HTML

Exemple primaire : index.php

```
<html>
  <head>
   <title>exemple basic</title>
  </head>
  <body>
<?php $nom="toto";
echo "hello $nom"; ?>
</body>
</html>
```

Après interprétation

```
<html>
  <head>
   <title>exemple basic</title>
  </head>
  <body>
hello toto
</body>
</html>
```

PHP : le langage

- C-like et perl-like :
  - Séparateur d' instruction **;**
  - Commentaires: // ou /* et */
  - variables préfixées par $
- La casse :
  - Les variables : OUI $nom et $Nom sont différents
  - Les fonctions : NON echo "bonjour"; ou eCHo"bonjour";
- Permet de définir des briques de base réutilisables
  (par inclusion de fichiers externes analysés comme du php+html)

  include "header.php";

  include_once "header.php";

  (pour éviter les inclusions multiples)

Les types

- Types supportés
  - Chaînes de caractères : "bonjour", ‘Ca va?’
  - Nombres entiers base 8,10,16 (O12 : base 8 , Ox12 : base 16)
  - Nombres à virgule flottante : -1.33e+4 vaut 13300
  - Booléen :true (ou 1) ou false(ou 0,ou0.0 ou vide ou NULL)
  - Tableaux (vecteurs ou associatifs, multidimensionnels)
  - Constantes : **define**("MAX",255);
  - Objets (POO)
- Pas de déclaration de type (l’affectation d’une variable détermine son type)

  $a = 1; // $a est un entier

  $a[0] = ‘xi’; // $a devient un tableau dont le 1-ier élément est une chaîne
- Transtypage (cast) à la C :

  $d = 9/2; // $d vaut 4.5

  $n = (int)$d; // $n vaut 4

Les variables

- En PHP, les variables sont représentées par un signe dollar "$" suivi du nom de la variable. Le nom est sensible à la casse (i.e. *$x != $X*).
- Un nom de variable valide doit commencer par une lettre ou un souligné (_), suivi de lettres, chiffres ou soulignés.(comme les autres entités PHP
- En PHP 3, les variables sont toujours assignées par valeur.
  - C'est-à-dire, lorsque vous assignez une expression à une variable, la valeur de l'expression est recopiée dans la variable. Cela signifie, par exemple, qu'après avoir assigné la valeur d'une variable à une autre, modifier l'une des variables n'aura pas d'effet sur l'autre.
- Depuis PHP 4, PHP permet aussi d'assigner les valeurs aux variables par référence.
  - la nouvelle variable ne fait que référencer (en d'autres terme, "devient un alias de", ou encore "pointe sur") la variable originale. Les modifications de la nouvelle variable affecteront l'ancienne et vice versa.
  - Pour assigner par référence, ajoutez simplement un & (ET commercial) au début de la variable qui est assignée

Les références

- Exemple 1

```
$foo = 5;            // Assigne la valeur 5 à $foo
$bar = &$foo;    // Référence $foo avec $bar.
$bar = 2 * $bar; // Modifie $bar ... et $foo!
// $foo vaut aussi 10
```

- Exemple 2

```
function incBad($var)   {   $var++;      }

function incOK(&$var)  {    $var++;     }

$nbr = 1;
incBad($nbr);  // passage de $nbr par valeur => $nbr inchangé
incOK($nbr);   // passage de $nbr par référence
echo $nbr;      // sa valeur a donc été modifiée : $nbr vaut 2
```

Fonctions d'affichage

Toutes ces fonctions écrivent sur la sortie standard:

- echo()
- print()
- printf($format, $arg1[, $arg2, …]) : écriture formatée comme en C, i.e. la chaîne de caractère $format contient le format d'affichage des variables passées en argument

Exemples :

```
print(''Bonjour $name'');
printf(''Bonjour %s'', $name);
```

Chaînes de caractères (1)

- Longueurs illimitées
- Délimiteurs :
  - Guillemets simples: pas d'interprétation sauf \\ et\' pour \ et ‘
  - Guillemets doubles: interprétation des variables et \n, \t, \r, \$, \\ et \’’
- Affichage par echo ou print (printf pour chaînes formatées comme en C)
- Opérateur de concaténation de chaînes : . (point)
- Exemples

```
$foo = 'Hello';
$bar = 'World';
echo $foo . ' ‘ . $bar;
echo "$foo $bar";
```

Chaînes de caractères (2)

- Caractères spéciaux
  - \n : saut de ligne dans le code généré mais pas en html… (</br>)
  - \r : retour à la ligne
  - \t : tabulation
  - \$: caractère $
- Quelques fonctions prédéfinies

strlen($str) : retourne le nombre de caractères

strtolower($str) : conversion en minuscules

trim($str) : suppression des espaces de début et de fin de chaîne

substr($str,$i,$j) : retourne une sous chaîne de $str de taille $j et débutant à la position $i

addslashes($str) : déspécialise les caractères spéciaux

Chaînes de caractères (3)

Syntaxe Here-doc

```
<<<MARQUE
Texte libre sur plusieurs lignes pouvant contenir
des variables et des doubles quotes
MARQUE
```

- Exemple

```
$strForm = <<<EOT
<form name="form" id="form" method="post"  action=" $scriptName">
 <table bgcolor="#dcdcdc" border="1" width="25%">
  <tbody>...
EOT;
echo $strForm;
```

- Avantages
  - Évaluation des variables (équivalent à une chaîne entourée de doubles quotes)
  - Pas besoin de protéger les doubles quotes

Tableaux (1)

- **Les tableaux**
  - **Ensemble d'associations clé/valeur** (Index :clé)
    - Clé = entier => vecteur
    - Clé = chaîne de caractères => tableau associatif (map)
  - **Création** à l'affectation
    - $tab[0]=125 ; clé entière, valeur entière
    - $tab[1]="toto" ; clé entière , valeur chaîne
    - $tab["titi"]="bonjour"; clé chaîne, valeur chaîne
    - $tab[1][3]="bonjour" ; tableau à deux dimensions
    - ….
  - **Création** par array ou list
    - $tab1 = array("one", "two"); # 0 =>"one" et 1 =>"two"
    - $tab2 = array("un"=>1, "deux"=>2); la clé est donnée à la déclaration
    - $mixture = array (
      "fruits" => array ("a"=>"orange", "b"=>"banane", "c"=>"pomme"),
      "nombres" => array (1, 2, 3, 4, 5, 6),
      "trous" => array ("premier", 5 => "second", "troisième")
      );

Tableaux (2)

- Nombre d'éléments d'un tableau : sizeof($tab) ou count($tab)

```
<?php
$tab = array (
"fruits" => array ("a"=>"orange", "b"=>"banane", "c"=>"pomme"),
"nombres" => array (1, 2, 3, 4, 5, 6),
"trous" => array ("premier", 5 => "second", "troisième"));
print_r($tab);echo "<br>\n";
echo "taille du tableau fruits= ".sizeof($tab["fruits"])."<br>";
echo "taille du tableau nombres= ".sizeof($tab["nombres"])."<br>";
?>
```

Donne :

```
Array ( [fruits] => Array ( [a] => orange [b] => banane [c] => pomme ) [nombres] => Array ( [0]
    => 1 [1] => 2 [2] => 3 [3] => 4 [4] => 5 [5] => 6 ) [trous] => Array ( [0] => premier [5] =>
    second [6] => troisième ) )
    taille du tableau fruits= 3
    taille du tableau nombres= 6
```

Tableaux (3)

- Suppression d' un élément unset($tab["toto"]);
- Tri d' un tableau: sur les clés et sur les valeurs
  - Conservation de l'association clé/valeur
    - asort et arsort : tri sur les valeurs (croissant ou reverse décroissant)
    - ksort et krsort : tri sur les clés
  - Perte de l'association
    - Sort : tri sur les valeurs et réassigne les clés 0,1,2,…
- Pointeur de tableau
  - à chaque tableau correspond un pointeur interne qui est une référence sur l'élément courant
    - current($tab) désigne l'élément courant (alias pos)
    - next($tab) déplace le pointeur vers l'élément suivant
    - prev($tab) déplace le pointeur vers l'élément précédent

    Ces 3 fonctions retournent la valeur pointée du tableau (ou false)
    - end($tab) déplace le pointeur sur le dernier élément
    - reset($tab) déplace le pointeur sur le premier élément
    - key($tab) donne la clé pour le pointeur

Tableaux (4)

- Soit le programme

```
<?$tab = array("b"=>1, "o"=>5, "n"=>8, "j"=>9,"u"=>2,"r"=>3);
echo "<h3>tableau initial :</h3> <BR>";
print_r($tab);echo "<br>";
$val = current($tab); $key= key($tab);
echo "1) key :$key valeur : $val<br>";
$val = next($tab); $key= key($tab);
 echo "2) key : $key  valeur : $val<br>";
arsort($tab);
echo "<h3>tableau après tri arsort :</h3> <BR>";
print_r($tab);echo "<br>";
 end($tab); prev($tab);  $key= key($tab);
 $val = prev($tab);
echo "3) key :$key, valeur : $val<br>";
?>
```

RESULTAT: Ah, l'ordre des prev et des key…

tableau initial :

Array ([b] => 1 [o] => 5 [n] => 8 [j] => 9 [u] => 2 [r] => 3)
1) key :b valeur : 1
2) key : o valeur : 5
tableau après tri arsort :

Array ([j] => 9 [n] => 8 [o] => 5 [r] => 3 [u] => 2 [b] => 1)
3) key :u, valeur : 3

Tableaux (5)

- Extraction des éléments d'un tableau
  - list($v1,$v2)=$tab: assigne la liste des variables
  - extract($tab) : extraction de toutes les valeurs, chaque valeur est recopiée dans une variable ayant pour nom la valeur de la clé
- Parcours des éléments d'un tableau (vecteur)
  - each($tab) retourne la paire clé/valeur courante et avance le pointeur, la paire est retournée dans un tableau de 4 éléments: 0=>clé,1=>valeur,key=>clé,value=>valeur
  - foreach(): permet de parcourir un tableau:

Exemple:

```
<?
$a = array(1, 2, 3, 17);
$i = 0;
foreach ($a as $v) {
   echo "\$a[$i] => $v <br>\n";
   $i++;
}
?>
```

```
$a[0] => 1
$a[1] => 2
$a[2] => 3
$a[3] => 17
```

Tableaux (6)

- Parcours des éléments d'un tableau associatif
  - each($tab) retourne la paire clé/valeur courante et avance le pointeur , la paire est retournée dans un tableau de 4 éléments: 0=>clé,1=>valeur,key=>clé,value=>valeur
  - foreach(): permet de parcourir un tableau:

Exemple:

```
<?
$a = array ( "one" => 1,
    "two" => 2,  "three" => 3);
foreach($a as $k => $v) {
       print "\$a[\"$k\"] => $v.\n";
}
?>
```

```
$a["one"] => 1
$a["two"] => 2
$a["three"] => 3
```

Tableaux (7)

- Tableaux multidimensionnels. Exemple :

```
$a = array();
$a[0][0] = "a";
$a[0][1] = "b";
$a[1][0] = "y";
$a[1][1] = "z";
foreach ($a as $v1) {
  echo "<b>";
  print_r($v1);echo "</b><br>";
  foreach ($v1 as $v2) {
      echo " $v2 <br>\n";
   }
}
```

Array ([0] => a [1] => b)
a
b
Array ([0] => y [1] => z)
y
z

Constantes

define(‘’ MAX ’’,255);

define (‘’ NOM ’’, ’’Berger ’’);

- Constantes prédéfinies par php
- Exemple:
  - **PHP_VERSION** (**string**)
  - **PHP_OS** (**string**)

  Le code :

  echo '<pre>';

  print_r(get_defined_constants());

  echo '</pre>';

  Vous donne la liste de celles-ci dont:

  [E_ERROR] => 1

  [E_WARNING] => 2

  [E_PARSE] => 4

  …

  [PHP_VERSION] => 4.3.3

  [PHP_OS] => WINNT

  [PHP_SAPI] => apache

  …. Dont les types d’erreurs

Les erreurs

- Les types

  [E_ERROR] => 1 erreur d'exécution
  [E_WARNING] => 2 alerte
  [E_PARSE] => 4 erreur d'analyse
  [E_NOTICE] => 8 notes (alertes ignorables…mais)
  ……

- Changement du niveau d'erreurs reportées

  Error_reporting(nnn) (par défaut souvent nnn=1+2+4)

- Envoyer un message d'erreur quelque part

  error_log (message, message_type, destination [,extra_headers])

  – *message_type*=0 : Par défaut dans le logger
  – *message_type*=1: par mail défini par *destination* avec *extra_headers* :subject,from, reply-to possibles
  – *message_type=2 :* remote_debugging connection php
  – *message_type=3* : ajout de ligne dans le fichier *destination*

Opérateurs (1)

- Opérateurs arithmétiques
  - addition : $a + $b
  - soustraction : $a - $b
  - multiplication : $a * $b
  - division : $a / $b
  - modulo (reste de la division entière) : $a % $b
- Concaténation de chaîne de caractère •
- Opérateurs binaires
  - ET bit à bit : $a & $b
  - OR bit à bit : $a | $b
  - XOR bit à bit : $a ^ $b
  - NON bit à bit : ~$a
  - décalage à droite de $b bits : $a >> $b (à gauche de $b bits : $a << $b)
- Opérateurs logiques
  - ET logique : and ou && (les deux sont possibles)
  - OU logique : or ou ||
  - XOR logique : xor
  - NON logique : !

Opérateurs(2)

- Opérateurs d'affectation
  - affectation avec le signe =

```
$n1 = ($n3 = 5) + 1; # $n3 vaut 5 et $n1 vaut 6
```

  - les opérateurs combinés : +=, -=, *=, /=, .=, &=,|=, ^=, <<=, >>=, ~=

```
$a += 1; # équivalent à $a = $a + 1;
$ch .= "!"; # équivalent à $ch = $ch . "!";
```

  ++ est équivalent à += 1, -- est équivalent à -= 1
- Opérateurs de comparaison
  - égal à : $a == $b
  - différent de : $a != $b
  - inférieur à : $a < $b et supérieur à : $a > $b
  - inférieur ou égal à : $a <= $b et supérieur ou égal à : $a >= $b
- Opérateur ternaire

  (condition) ? (expr1) : (expr2);

  renvoie expr1 si condition est vraie sinon renvoie expr2

```
$max = ($a >= $b) ? $a : $b; # $max donne le  max($a,$b)
```

Structures de contrôle conditionnelles (1)

- **if** (condition) { commande1;commande2;}
- if (condition) {
  commande1;commande2;
  } else {
  commande3;commande4;
  }
- elseif :

```
if (condition1) {
        # si condition1 est vraie
    } elseif (condition2) {
        # si condition2 est vraie (et pas condition1)
    } elseif (condition3) {
        # si condition3 est vraie (et ni condition1, ni condition2)
    } else {
        # si ni condition1, ni condition2, ni condition3 ne sont vraies
    }
```

Structures de contrôle conditionnelles(2)

- Le **switch** comme en C

```
switch(expr) {
  case (valeur1) :
  # à exécuter si expr vaut valeur1
  break;
  case (valeur2) :
  # à exécuter si expr vaut valeur2
  break;
  default :
  # à exécuter dans tous les autres cas
}
```

Structures de contrôle : boucles tant que

- Boucle "tant que"

```
while (condition) {
# exécuté tant que la condition est vraie
}
```

ou

```
while (condition) :
# exécuté tant que la condition est vraie
endwhile;
```

- Boucle "do…while"

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

Structures de contrôle : boucles for

- Boucle "for"

```
for ($i = 1; $i <= 100; $i++) {
# exécuté tant que ($i <= 100) est vraie
}
```

- ou

```
for ($i = 1; $i <= 100; $i++) :
# exécuté tant que ($i <= 100) est vraie
endfor;
```

- L'instruction break permet de sortir d'une boucle
- L'instruction continue permet de passer directement à l'itération suivante de la boucle

Structures de contrôle : boucles foreach (1)

(Parcours des tableaux simples)

```
foreach ( $values as $value ) {
   // on récupère dans $value chaque élément de
   $values
}
```

Exemple:

```
<?
$a = array(1, 2, 3, 17);
$i = 0;
foreach ($a as $v) {
   echo "\$a[$i] => $v <br>\n";
   $i++;
}
?>
```

```
$a[0] => 1
$a[1] => 2
$a[2] => 3
$a[3] => 17
```

Structures de contrôle : boucles foreach (2)

(Parcours des tableaux associatifs)

- Boucle "foreach"

```
foreach ( $_SESSION as $key => $value ) {
    // on récupère les attributs de la session
}
```

- Équivalent à :

```
while ( list($key, $value) = each($_SESSION) ) {
    // on récupère les attributs de la session
}
```

Variables prédéfinies (1)

- Variables prédéfinies - tableaux globaux PHP4.1.0
  - $GLOBALS : ce tableau contient toutes les variables globales définies.
  - $_SERVER : ce tableau contient toutes les variables fournies par le serveur Web ou le client.*
  - $_GET : ce tableau contient toutes les variables provenant de l'URL courante.
  - $_POST : ce tableau contient toutes les variables provenant d'un formulaire en méthode **post**.
  - $_COOKIE : ce tableau contient toutes les valeurs et noms des cookies envoyés par le client.

Variables prédéfinies (2)

- Variables prédéfinies - tableaux globaux PHP4.1.0 (suite)
  - $_FILES : ce tableau contient les variables fournies par le navigateur lors d'un upload de fichier par le client.
  - $_ENV : ce tableau contient toutes les variables fournies par l'environnement PHP.
  - $_REQUEST : ce tableau contient toutes les variables fournies par l'intermédiaire d'un script d'entrée (GET, POST, COOKIE... par exemple).
  - $_SESSION : ce tableau contient toutes les variables de session utilisées

Formulaire HTML et $_REQUEST

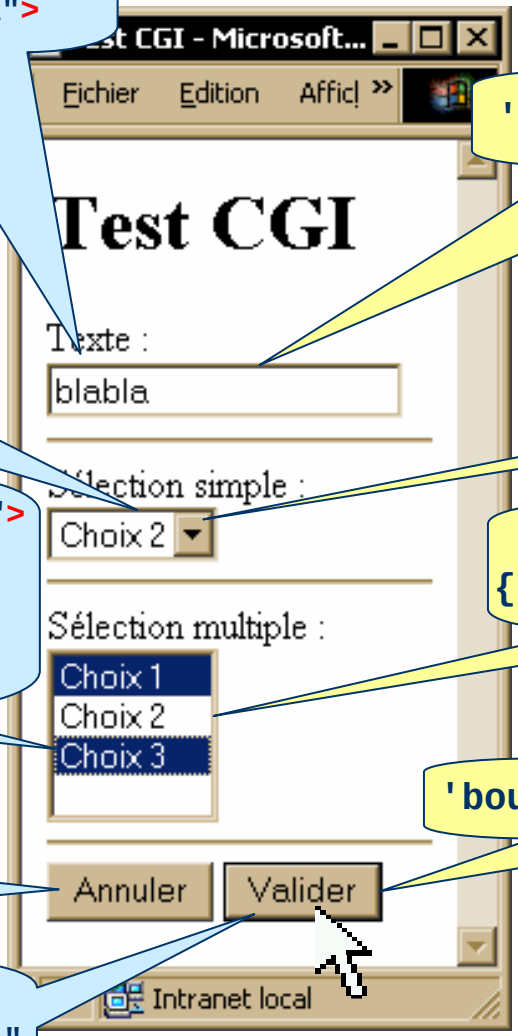
```
<input type="text"
       name="texte_court"
       value="blabla">
```

```
<select name="sel_simple">
  <option value="1">Choix 1</option>
  <option value="2">Choix 2</option>
  <option value="3">Choix 3</option>
</select>
```

```
<select multiple name="sel_multiple[]">
  <option value="1">Choix 1</option>
  <option value="2">Choix 2</option>
  <option value="3">Choix 3</option>
</select>
```

```
<input type="reset"
       name="bouton_reset"
       value="Annuler">
```

```
<input type="submit"
       name="bouton_submit"
       value="Valider">
```




```
$_REQUEST
```

```
'texte_court' => "blabla"
```

```
'sel_simple' => "2"
```

```
'sel_multiple' =>
{ [0] => "1", [1] => "3" }
```

```
'bouton_submit' => "Valider"
```

Récupération des données d'un formulaire (1)


```
<form action="editMember1Action.php" method="post">
<table width=300>
 <tr><td>Civilité :</td>
 <td><input type=radio value="Mr" name=civilite>Mr
    <input type=radio value="Mme" name=civilite>Mme
    <input type=radio value="Mlle" name=civilite>Mlle
 </td></tr>
 <tr><td>Nom</td>
    <td><input type=text size=20 name=nom></td>
 </tr>
 <tr><td>prénom</td>
   <td><input type=text size=20 name=prenom></td></tr>
 <tr><th colspan=2><input type=submit value="Envoyer"></th></tr>
```

Récupération des données d'un formulaire (2)

editMember1Action.php

```
<?php
//récupération des données
$civilite=$_POST["civilite"];
$nom=$_POST["nom"];
$prenom=$_POST["prenom"];
?>
Bonjour <?echo $civilite;?>
 <b> <?echo strtoupper($nom);?></b>
 <?echo ucfirst($prenom);?>
```

Affiche

Bonjour Mr **NOAH** Yannick

Récupération des données d'un formulaire (3)


```
<form method="POST"
   action="<?php  echo $_SERVER['PHP_SELF']; ?>" >
Comment avez-vous trouvé ce cours ?
<input type="radio" name="note" value="super" CHECKED >Super
<input type="radio" name="note" value="bien">Bien
<input type="radio" name="note" value="hasard">Par hasard

<br>Que prendrez vous en dessert ?
<input type="checkbox" name="dessert[]" value="pomme">Pomme
<input type="checkbox" name="dessert[]" value="poire">Poire
<input type="checkbox" name="dessert[]" value="orange">Orange
<br><input type="submit" value="Valider">
</form>
```

Récupération des données d'un formulaire (4)

```
<?php
// print_r($_REQUEST);
echo 'Vous avez trouvé ce cours ' . $_REQUEST['note'];
if ( isset($_REQUEST['dessert']) ) {
  $desserts = $_REQUEST['dessert'];
  echo " <br>Vous avez choisi: ";
  foreach ( $desserts as $dessert ) {
     echo $dessert . ' ';
  }
} ?>
```

Affiche

Vous avez trouvé ce cours super
Vous avez choisi: pomme poire