

Technologies Web

PHP

Marco Winckler

ICS-IRIT

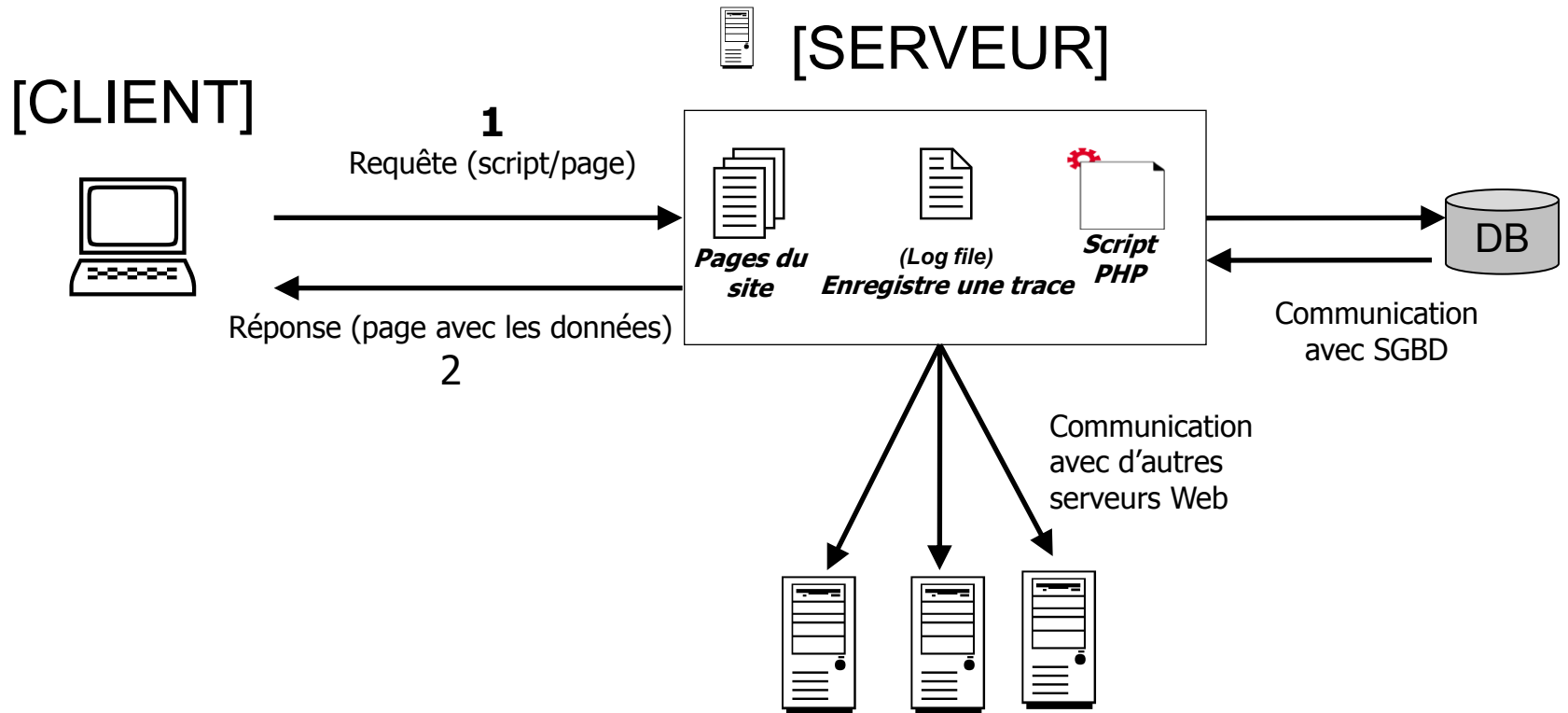
Université Paul Sabatier (Toulouse 3)

<http://www.irit.fr/~Marco.Winckler>

winckler@irit.fr



L'architecture Web



Exemples

Fichier exple1.php

```
<?PHP
echo "<H1>Cours de
      PHP</H1><br>";
echo "Voici un premier
      exemple de mise en
      oeuvre de PHP";
?>
```

Fichier exple2.html

```
<HTML>
    <HEAD>        </HEAD>
<BODY>
lancer le script PHP en cliquant <A
    href=http://127.0.0.1/exple1.php
    >ici</A>.
</BODY>
</HTML>
```

Exemples (suite ...)

Fichier exple3.php

```
<HTML> <BODY>
  <H1>Hello</H1>
  <?PHP
    for ($i=1;$i<10;$i++) {
      for ($j=1; $j<$i; $j++) { echo "*"; }
      echo "<BR>";
    }
  ?>
```

Voila, c'est le dernier exemple d'utilisation de PHP

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

Exemples : résumé

- 3 façons d'utiliser PHP
 - Donner au navigateur l'URL d'un fichier ne contenant QUE du PHP
 - Permettre l'accès à un fichier PHP par l'intermédiaire d'un lien (*balise A + attribut HREF*)
 - Ajouter du code PHP à l'intérieur de code HTML
 - On trouvera donc dans le fichier les balises HTML (<HTML>, </HTML>, ...) et PHP (<?php ... ?>)
- Extension d'un fichier contenant du code PHP
 - .php chaque fois que `<?php ... ?>` se trouve dans le fichier !

Notions de base

- Commentaires
 - `//` : pour mettre une ligne en commentaire
 - `/* ... */` : pour mettre un ensemble de lignes en commentaire
- Insertion de code PHP

```
<?php
... instructions PHP ...
?>
```
- Visibilité du code produit par PHP (affichage / source)
 - Uniquement le code HTML produit (*même si IF, un FOR*)
 - Aucune instruction PHP !
 - Sur une seule ligne, sauf utilisation de caractères spéciaux (`\n`)

Variables, types et opérateurs

- Typage des variables implicite en php
- Nom de variable TOUJOURS précédé de « \$ » (dollars)
- Portée des variables limitée au bloc contenant la déclaration
 - Variable globale : visible et modifiable dans tout bloc <?php ... ?> mais pas dans les fonctions
 - Variable locale : déclarée, visible et modifiable dans une fonction
- Types existants :
 - entier (int), réel (double),
 - chaîne de caractères (string), tableau (array),
 - objet (object), booléen (boolean).
- Typage explicite par *cast*
 - *Exemple :*
 - \$str = "12"; // \$str vaut la chaîne "12"
 - \$nbr = (int)\$str; // \$nbr vaut le nombre 12

Variables, types et opérateurs

- Fonctions sur les variables :
 - Très utile :
 - `isset($var)` : renvoie vrai si la variable existe
 - `define(nom, valeur)` : associe un nom symbolique à une valeur décimale
 - A connaître :
 - `empty($var)` : renvoie vrai si la variable est vide
 - `unset($var)` : détruit une variable
 - `gettype($var)` : retourne le type de la variable
 - `settype($var, "type")` : convertit la variable en type *type* (cast)
 - A comprendre :
 - `is_long()`, `is_double()`, `is_string()`, `is_array()`, `is_object()`, `is_bool()`, `is_float()`, `is_numeric()`, `is_integer()`, `is_int()`...

Variables, types et opérateurs

- Opérateurs arithmétiques
 - + (addition), - (soustraction), * (multiplié), / (divisé), % (modulo), ++ (incrément), --(décrément)
- Opérateurs d'assignement
 - = (affectation),
 - *= ($\$x*=\y équivalent à $\$x=\$x*\$y$), /=, +=, -=, %=
- Opérateurs logiques
 - and, && (et), or, || (ou), xor (ou exclusif), ! (non)
(and et or sont de priorité plus faibles)
- Opérateurs de comparaison
 - == (égalité), < (inférieur strict), <= (inférieur large)
 - >, >=, != (différence)

Les chaînes de caractères

- Encadrée par " " ou ' '
 - " " : permettent l'évaluation des variables et caractères spéciaux contenus dans la chaîne
 - ' ' : pas d'évaluation des variables, caractères spéciaux
- Caractères spéciaux :
 - \n (nouvelle ligne), \r (retour à la ligne)
\t (tabulation horizontale), \\ (antislash),
\\$ (caractère dollars), \" (double quote)
 - Impact sur le code HTML produit
(pas sur la visualisation dans le navigateur)

Les chaînes de caractères

- Opérateurs de concaténation : `.`
 - `$toto = "bonjour" . " tous le monde !"`
- Autres fonctions
 - Très utiles :
 - `strlen($str)` : retourne le nombre de caractères d'une chaîne
 - `strtolower($str)` : conversion en minuscules
 - `strtoupper($str)` : conversion en majuscules
 - A comprendre :
 - `trim($str)` : suppression des espaces de début et de fin de chaîne
 - `substr($str,$i,$j)` : retourne une sous chaîne (entre les positions i et j)
 - `strnatcmp($str1,$str2)` : comparaison de 2 chaînes
 - `addslashes($str)` : déspecialise les caractères spéciaux (`'`, `"`, `\`)
 - `ord($char)` : retourne la valeur ASCII du caractère

Entrées / Sorties

- `echo()` : écriture dans le navigateur
- `print()` : écriture dans le navigateur
- `printf([$format, $arg1, $arg2])` : écriture formatée, i.e. la chaîne de caractère est constante et contient le format d'affichage des variables passées en argument
- *Exemples :*
 - `echo "Bonjour $name";`
 - `print("Bonjour $name");`
 - `printf("Bonjour %s", $name);`
- Fichier et BD : plus tard ...

Les tableaux nominatifs (simples)

- Type array : contient des éléments de tout type
- Initialisation statique : séparation des valeurs par des ,
 - *Exemple :*
 - `$tab_colors = array('red', 'yellow', 'blue', 'white');`
 - `$tab = array('foobar', 2002, 20.5, $name);`
- Initialisation dynamique : utilisation des [] (avec ou sans indice)
 - *Exemples :*
 - `$note[] = 15;` OU `$villes[0] = "Paris";`
 - `$note[] = 18;` `$villes[1] = "Londres";`
 - `$note[] = 13;` `$villes[2] = "Lisbonne";`
 - Le premier indice est le 0 (comme en Java ou en C)
- Utilisation
 - `echo $note[10];` `// pour accéder à la 11ème note`

Les tableaux nominatifs

- Parcours d'un tableau : *\$tab* = array(12,15,7,18);
 - FOR + count(*\$tab*)
for (*\$i*=0; *\$i*<=count(*\$tab*); *\$i*++)
{ echo "*\$tab*[*\$i*]
"; }
 - FOREACH *\$tab* AS *\$var*
foreach (*\$tab* as *\$uneNote*)
{ echo "*\$uneNote* \n "; }
 - La variable *\$uneNote* prend successivement (à chaque tour de boucle) toutes les valeurs du tableau *\$tab*.
 - Impossible de récupérer l'indice
 - reset(*\$tab*), next(*\$tab*), current(*\$tab*), key(*\$tab*)

Les tableaux nominatifs

- Parcours d'un tableau (suite ...)
 - `$var = each($tab)` : renvoie un tableau contenant l'élément courant du tableau (indice + valeur) et avance sur l'élément suivant
 - Nécessité du reset avant première utilisation
 - Accès à la valeur via : `current($var)`
 - Accès à l'indice par : `key($var)`
 - Souvent couplé à un *while*
 - `list($ind, $var) = each($tab)` : en plus du comportement du `each`, cette instruction stocke l'indice dans `$ind` et la valeur dans `$var`
 - Accès à la valeur directement par `$var`
 - Souvent couplé à un *while*

Les tableaux nominatifs

- Fonctions de manipulation des tableaux :
 - Très utiles :
 - `count($tab)` : retournent le nombre d'éléments du tableau
 - `in_array($var,$tab)` : *true* si la valeur de `$var` est dans le tableau `$tab`
 - `list($var1,$var2...)` : transforme un tableau en liste de variables
 - `sort($tab)` : trie alphanumérique les éléments du tableau
 - `implode/join($tab, $str)` : retournent une chaîne de caractères contenant les éléments du tableau `$tab` joints par la chaîne `$str` (*n'affiche que les valeurs ... pas les clés : cf plus loin*)
 - `serialize($tab)` : retourne une chaîne de caractère représentant la structure et le contenu d'un tableau (y compris clés et valeurs)
 - `explode($delim,$str)` : retourne un tableau dont les éléments résultent du découpage de la chaîne `$str` par le délimiteur `$delim`
 - A comprendre :
 - `range($i,$j)` : retourne un tableau contenant un intervalle de valeurs
 - `shuffle($tab)` : mélange les éléments d'un tableau
 - `rsort($tab)` : trie alphanumérique inverse les éléments du tableau
 - `array_merge($tab1,$tab2,$tab3...)` : concatène les tableaux passés en arguments
 - `array_rand($tab)` : retourne un élément du tableau au hasard
 - `usort ($tab, "fonction")` : Trie les éléments grâce à la fonction *fonction* définie par vous qui doit prendre 2 paramètres et retourner un entier, qui sera inférieur, égal ou supérieur à zéro suivant que le premier argument est considéré comme plus petit, égal ou plus grand que le second argument.

Les tableaux associatifs

- Tableaux associatifs (ou *hashtable*)
 - Liste de couples (clé, valeur)
 - Initialisation statique : `array("clé1" => valeur1, ...)`
`$personne = array("Nom"=>"César","Prénom"=>"Jules",
"age"=>25);`
 - Initialisation dynamique : `$tab["cle1"] = valeur1`
`$personne["Nom"] = "César";`
`$personne["Prénom"] = "Jules";`

Les tableaux associatifs

- Parcours de tableaux associatifs
 - for : **pas pratique**
 - foreach (*\$tabA* as *\$k* => *\$v*) ... :
 - stocke successivement la clé dans *\$k* et la valeur dans *\$v*
 - list (*\$key*, *\$var*) = each(*\$tab*)
 - key(*\$tab*) / current(*\$tab*)
 - Retourne la clé/valeur de l'élément courant du tableau
 - Utiliser reset(*\$tabA*) au début, puis next(*\$tabA*)
 - Incompatible avec each, seulement utilisable avec next !

Structures de contrôle

- Structures conditionnelles

- IF

```
if( ... )  
{ instructions; }  
elseif (...)  
    { instructions; }  
else  
    { instructions; }
```

- SWITCH

```
switch( ... )  
{  
    case valeur : { instructions; } break;  
    case ...  
    default : { ... }  
}
```

Structures de contrôle

- Itérations

- FOR : nombre d'itérations connu

- for(initialisation ; conditions de poursuite ;
incrémentation)*
{ instructions; }

- WHILE : nombre d'itérations indéfini /
conditionnel

- while(expression booléenne)*
{ instructions; }

- *foreach ... in ...*

Fonctions

- Fonctions = sous-programmes
 - Décompose le code en sous-parties
- Déclaration
 - `function nomFonction ($par1, $par2, ...) { ...; return ...; }`
`function` `mafonction` (`$toto`, `$nom`)
`{`
 `$toto += 15; echo "Salut $nom !";`
 `return` (`$toto+10`); // affiche "salut machin !" et retourne `toto+25`
`}`
- Utilisation
 - `$resultat = nomFonction ($v1, 39, $v3, ...)`
 `$i = 20; $res = mafonction ($i, "Winckler");`
 // "Salut Winckler !" est affiché et res vaut 45

Fonctions

- Compléments :
 - static *\$var* : variable de fonction dont la valeur reste sauvegardée jusqu'au prochain appel de la fonction
 - global *\$var* : variable visible et modifiable dans la fonction mais déclarée dans le code comme globale au programme
 - \$GLOBALS : tableau associatif des variables globales
 - Valeur par défaut aux paramètres

```
function toto($par1 , $par2="val_par_defaut")
```
 - Passage **forcé** de paramètres par référence

```
function toto(&$v) {$v+=100;}
```

Appel : `$x=5; toto($x); echo "$x"; // affiche 105`

Fonctions

- Compléments (suite ...) :
 - Et si une fonction retourne plusieurs résultats ???
 - Paramètres passés par adresse (référence)
 - Variable de retour (de la fonction) de type ARRAY

Inclusion d'un autre fichier dans un script PHP

- **require**
 - Insert dans le code le contenu du fichier spécifié même si ce n'est pas du code php.
- **include**
 - Évalue et insert à chaque appel (même dans une boucle) le contenu du fichier passé en argument.

Miscellanées

- `die("msg")`
 - Provoque l'arrêt prématuré du script PHP et affiche *msg*
- `exit()`
 - Idem sans affichage de message
- `
` vs. `\n`
 - `
` : force le passage à la ligne dans le navigateur
 - `\n` : force le passage à la ligne dans le fichier HTML résultant de l'exécution du script PHP

Plan de la suite

- Après PHP vu comme un langage de programmation, voici quelques aspects de PHP qui justifient pleinement son existence !
 - Communication PHP / Pages Web
 - Formulaires, URL, Cookie, Session
 - BD
 - Manipulation de fichiers

Communications Pages Web - PHP

- Objectifs : passer des paramètres d'une page Web vers un script (ou l'inverse)
- Origine des données : les formulaires
- Méthodes de passage de paramètres
 - Méthodes POST du formulaire
 - Variables d'environnement
 - Méthode GET du formulaire et URL
 - Cookie
 - Variables de sessions

Rappel : formulaires HTML

Éléments de base

- Trois catégories :
 - INPUT : différents types d'interacteurs permettant d'entrer des données (de l'utilisateur vers le système) :
 - Champ de texte, champ de texte caché,
 - Bouton de soumission, de remise à zéro, de lancement d'un script
 - SELECT : interacteurs s'appliquant à des ensembles de données (menu, liste)
 - Case à cocher, bouton radio
 - Menu déroulant
 - Liste
 - TEXTAREA : zone de saisie de texte libre
 - Texte long

Balises de formulaires

- **<FORM>**

- Marque le début de la description d'un formulaire
- Entre <FORM> et </FORM> se trouvera la définition des boutons, listes, ...
- Attributes, propriétés, méthodes, évènements
- Attributs (caractérisent le script CGI qui sera appelé)
 - NAME = "nom_donné_au_formulaire"
 - METHOD = GET ou POST
 - ACTION = "le_nom_du_script_PHP_à_exécuter"
 - TARGET = fenêtre ou frame dans laquelle doit s'afficher le résultat du script CGI (une page Web)
 - Exemple

```
<FORM name=forme method=get action=/cgi-bin/annuaire>
```

Balises des composants

- **<INPUT>** ou **<INPUT type=text>**
 - Création d'un champs de texte vierge (scrolling automatique)
 - Attributs
 - NAME : nom du champ (pour utilisation dans les JavaScript et CGI)
 - SIZE : longueur de texte visible (en caractères)
 - MAXLENGTH : nb max de caracteres saisis dans le champs de texte
 - VALUE : préinscription d'une valeur dans le champ texte
 - Exemple :

```
<INPUT name="resultat" size=12 value="bonjour">
```
- **<INPUT type=password>**
 - Attributs identiques à l'objet texte
 - MAIS les caractères saisis apparaissent sous forme de *

Balises des composants

- `<INPUT type="button">`
 - Déclenche l'appel d'une fonction (*JavaScript par exemple*)
 - Ne transmet aucune information à un script CGI.
 - Attributs : NAME, VALUE
- `<INPUT type=submit>`
 - Définit un bouton qui déclenche l'envoi de tous les champs du formulaire vers le script CGI (indiqué dans l'attribut ACTION de `<FORM>`)
 - Attribut NAME : nom du bouton, omissible si un seul bouton SUBMIT
 - Attribut VALUE : spécifie le texte inscrit sur le bouton
- `<INPUT type="reset">`
 - Efface toutes les saisies et rétablit les valeurs par défaut de tous les éléments du formulaire
 - Attributs : NAME, VALUE

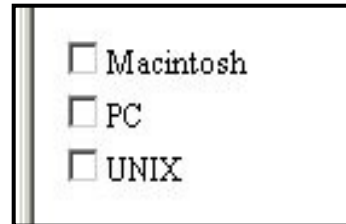
Balises des composants

- **<INPUT type=checkbox>**

- Définit un ensemble de cases à cocher permettant un choix multiple

- Exemple :

```
<INPUT type="checkbox" name="machine" value="mac">Macintosh<Br>  
<INPUT type="checkbox" name="machine" value="pc">PC<Br>  
<INPUT type="checkbox" name="machine" value="unix">UNIX<Br>
```



☐ Macintosh
☐ PC
☐ UNIX

- Attributs

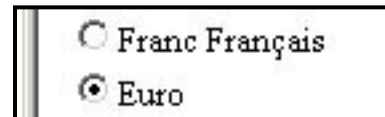
- VALUE : valeur transmise au script
 - NAME : nom de l'interacteur (identique ou différents pour chaque box)
 - CHECKED : sélectionné par défaut au chargement du document

- **<INPUT type=radio>**

- Définit un ensemble de boutons radio permettant un choix exclusif

- Exemple :

```
<INPUT type="radio" name="monnaie" value="franc">Franc Français<Br>  
<INPUT type="radio" name="monnaie" value="euro" checked>Euro<Br>
```



☐ Franc Français
☒ Euro

- Attributs NAME, CHECKED : identique à CHECKBOX

Balises des composants

- **<SELECT>**

- Marque le début d'une liste ou d'un menu déroulant

- Attributs

- NAME : nom de l'interacteur à l'intérieur du formulaire
 - SIZE : si > 1 alors création d'une liste, sinon création d'un menu
 - MULTIPLE : autorise une sélection multiple (*si SIZE > 1*)

- **<OPTION>**

- Permet l'ajout d'un nouvel item dans la liste ou le menu

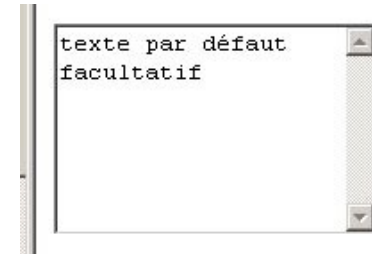
- Attributs

- VALUE, SELECTED

Balises des composants (5)

- **<TEXTAREA>**

- Définit une zone de texte
- Attributs, propriétés, événement, méthodes identique à `<INPUT type=text>`
- Attributs spécifiques
 - ROWS : nombre de lignes visibles (*= "7" dans l'exemple*)
 - COLS: nombre de colonnes visibles (*= "20" dans l'exemple*)
 - Texte par défaut : placé entre `<TEXTAREA>` et `>/TEXTAREA>`



Balises des composants

- `<INPUT type="hidden">`
 - Permet de cacher un champ de classe INPUT (invisible dans le browser)
 - Intérêt :
 - Conserver des valeurs sans les afficher dans les pages HTML, mais pouvant être utilisées dans les JavaScript et transmises à des CGI / PHP
 - Exemple : formulaires en plusieurs pages HTML, dont les dernières nécessitent des données acquises dans les pages précédentes pour exécuter un script CGI / PHP
 - Le numéro de compte bancaire d'un client est requis pour se connecter
 - Il est aussi requis après lors de l'édition d'un relevé de compte

PHP: Méthode POST

- Utiliser les valeurs des attributs NAME (des interacteurs du formulaire HTML) dans `$_POST["..."]` dans le script PHP souhaitant manipuler ces données
 - Le nom du script à lancer est indiqué par l'attribut ACTION de la balise FORM
 - Une variable portant le nom du bouton Submit utilisé est créée automatiquement
- Conséquences pour les noms à donner aux interacteurs (dans le source HTML)
 - text, button, textarea, radio : *nom*
 - checkbox, select (*multiple*) : *nom[]* (plusieurs réponses possibles => stockage dans un tableau des valeurs correspondant aux éléments sélectionnés)

Méthode POST : exemple

```
<!-- HTML -->
<form method="post" action="form1.php">
Entrez votre nom <input type=text
    name=nom size=20><br>
De quel instrument jouez vous ? <br>
<input type=checkbox name=instrument[]
    value=sax>saxo<br>
<input type=checkbox name=instrument[]
    value=clar>clarinette<br>
<input type=checkbox name=instrument[]
    value=bass>basse<br>
<input type="submit" value=" OK ">
</form>
```

```
// fichier reponseFormPOST.php
<?php

echo "bonjour <B>" . $_POST["nom"] .
    "</B><br>";
if (count($_POST["instrument"])!=0)
{ echo "vous jouez de : <UL>";
  foreach ($_POST["instrument"]as $instr)
  { if ($instr != "")
    { echo "<LI>$instr</LI>"; }
  }
  echo "</UL>";
}
else
{ echo "vous n'êtes pas musicien"; }

?>
```

IMPORTANT

- Ne pas oublier les [] si l'interacteur peut transmettre plusieurs valeurs (checkbox, list à sélection multiple)
- Valeur transmise au script PHP :
`$_POST["l'attribut NAME "]`
- Method = POST → Pas de valeur dans l'URL !

Variables d'environnement

- Deux variables PHP sont prédéfinies :
 - `$_GET` (ou `$HTTP_GET_VARS`) : tableaux associatifs contenant les noms et valeurs des variables envoyées par la méthode **GET**
 - `$_POST` (ou `$HTTP_POST_VARS`) : tableaux associatifs contenant les noms et valeurs des variables envoyées par la méthode **POST**
 - Rq : la clé contient la valeur exacte de l'attribut NAME de l'interacteur (pas de préfixe \$), la valeur pourra être de type ARRAY (tester avec `is_array($variable)`)
- **`$REQUEST_METHOD`** :
 - variable PHP prédéfinie valant **POST** ou **GET**
- Début de script PHP classique

```
if ($REQUEST_METHOD == "POST" )  
    { echo ("POST<br>"); ... }  
else { echo ("GET<br>"); ... }
```


PHP: Méthode GET

- Utilisation strictement identique au cas de la méthode POST
- Comportement subtilement différent que l'on va pouvoir réutiliser

PHP : URL

- Constat :
 - GET ajoute une chaîne de caractères à l'URL
 - La chaîne de caractères contient le nom et les valeurs de variables du formulaire
- Idée :
 - Utiliser cette chaîne de caractères
- Remarque :
 - Les couples nom/valeur sont séparés par des &

PHP : URL

- `$_SERVER["QUERY_STRING"]`
 - Variable PHP contenant les couples nom/valeur ajoutés à l'URL
- `$tab = split($car, $str)`
 - Fonction PHP découpant la chaîne *str* en sous-chaînes
 - Les sous-chaînes sont initialement séparées par *car* dans *str*
 - Chaque sous-chaîne constitue un élément du tableau *tab*
- `$v1 = urldecode($v2) / $v3 = urlencode($v4)`
 - `urldecode` : Décode une chaîne de caractère passée par URL (*v2*)
 - `urlencode` : code une chaîne de caractère (*v4*) pour qu'elle puisse être passée par une URL
 - Rq : requis pour décoder/encoder les espaces, accents, [, ...
- Ou alors utiliser `$_GET` : même fonctionnement que `$_POST`

PHP : URL

- Autre idée liée à la première méthode d'utilisation de GET :
 - Établir un lien vers un script PHP ()
 - Ajouter à l'URL un ensemble de couples nom de variable/valeur
- Intérêt :
 - Echanger des paramètres sans formulaire
- Application :
 - Gestion des menus d'une page web en PHP

PHP : URL

- Processus d'utilisation des données passées par l'URL
 - Récupérer la chaîne de caractères
 - `$_SERVER["QUERY_STRING"]`
 - Identifier les chaînes de caractères "*nom=valeur*"
 - `$liste = Split("&", ...)`
 - Pour chaque chaînes de caractères, séparer le nom et la valeur de la variable
 - ```
While (list($ind, $c) = each ($liste)
 list($var, $val) = split ("=", $c)
```
  - Enfin, `urldecode`
  - ... ou alors directement `$_GET["l'attribut NAME"]`

# PHP : Sessions

- PHP n'a pas de mémoire !
  - Variables définies dans une page, inconnue dans les autres pages du site.
- Mécanisme de session
  - Permet de sauvegarder des variables pendant un certain temps (2h par défaut) sur le serveur tandis qu'un identifiant de session est posté sous la forme d'un cookie chez le client (ou via l'URL si le client refuse les cookies)
  - But : espionner, suivre, analyser ... le parcours d'un internaute sur un site web

# PHP : Sessions

- Principales fonctions
  - `session_start()` : démarre une session
  - `session_destroy()` : détruit les données de session et ferme la session
  - `session_register("var")` : enregistre la variable `$var` dans la session en cours, **attention**, ne pas mettre le signe \$ (dollars) devant le nom de variable
  - `session_unregister("var")` : détruit la variable `$var` de la session en cours

# PHP : Sessions

- Principe de fonctionnement
  - Attribution d'un identificateur de session unique pendant la durée de visite du site : (**\$PHPSESSID**)
  - Création, stockage et manipulation de variables de sessions (*globales à la navigation dans le site*)
  - A la fin de la session, toutes les variables de session sont sauvegardées sur le serveur et pourront être redéployées (similaire au cookies mais sur le serveur)



# PHP : Sessions

- Propagation des variables de session
  - Par le biais de cookies (par défaut)
    - Création du fichier "sess\_".\$PHPSESSID  
par exemple : *page/i:41;*
  - Via l'URL
    - Utile si les cookies ne sont pas accepté
    - Exple : <a href=session.php?<?=SID?>> clic </A>
  - Dans les deux cas, les variables de sessions s'utilisent comme des variables globales d'un script PHP.
- MAIS ...
  - Fonctionnement des sessions souvent dépendant de la configuration du serveur et/ou client ...

# PHP : Cookies

- Cookies :
  - fichier texte que le serveur écrit sur le disque dur du client
- Fonctions :
  - `set_cookies()` : écrit un cookie sur le client
  - `empty($nomCookies)` : *true* si le cookie n'existe pas.
  - Rq : lorsque le client se connecte à un serveur pour charger une page web, si un cookies relatif à cette page web existe sur le client, il est automatiquement envoyé au serveur
    - ➔ Les cookies sont stockés dans le tableau associatif `HTTP_COOKIE_VARS` sur le serveur (nom => valeur)

# PHP : Cookies

- Envoi d'un cookie
  - int **setcookie** ( *nom* [, *valeur* [, *date d'exp.* [, *path* [, *domaine* [, *sécurité* ]]]]] )
    - Champs entre [ ] = optionnels
    - Généralement : que les deux premiers paramètres
      - ➔ le cookie est simplement stocké jusqu'à fermeture du navigateur
        - Nom = nom du cookie : empty(\$nom) vaut donc true.
    - *Date d'exp.* : force le cookie à être écrit sur le client jusqu'à la date d'expiration ( par. *exple* : "*time()* + 10" = 10 secondes)
    - A écrire avant tout affichage dans le navigateur
- Intérêts des cookies
  - Contrôle du temps de connexion, passage de paramètres entre pages web
  - Très utiles pour les sessions

# PHP et MySQL

- Rappels
  - MySQL = langage d'interrogation de bases de données
  - Base de données = ensemble de tables
  - Table = ensemble d'enregistrements respectant un même profil
  - Profil = ensemble d'attributs (clés)
  - Enregistrement = ensemble de valeurs (une par attribut)
- Gestion, manipulation
  - via le langage MySQL
  - Via phpMyAdmin (version graphique du langage)

# PHP et MySQL

- Communication PHP – MySQL (fonctions les plus fréquentes)
  - `mysql_create_db` : crée une base de données (\*)
  - `mysql_drop_db` : supprime une base de données (\*)  
(\*) : actions gérables via PHPMyAdmin
  - `mysql_connect` (\$server,\$user,\$password) : établit une connexion avec le serveur MySQL
    - `my_sql_pconnect` : idem, mais version persistante, i.e. la connexion n'est pas close à l'issue du script (pas besoin de réouvrir la connexion à chaque script)
  - `mysql_select_db` : sélectionne une base de données
  - `mysql_query` : envoie une requête à MySQL
  - `mysql_error` : renvoie le texte du message d'erreur de la dernière opération
- Faire précéder l'appel de ces fonctions par @ bloque l'affichage d'un message d'erreur généré par MySQL

# PHP et MySQL : Création

- Création et initialisation de BD avec PHP
  - Fonction CREATE
    - `create database` *nomBD* : crée une BD
    - `create table` *nomT* : crée une table
  - Processus de création d'une table :
    - Connexion à MySQL
      - `$DB_id = mysql_connect( $serveur, $nom, $passwd)`
    - Création d'une base
      - `mysql_create_db( $nomDB)`
    - Sélection de la base dans laquelle créer la table
      - `mysql_select_db( $nomDB, $DB_Id)`
    - Création de la table
      - `mysql_query( $requete, $DB_ID)`

# PHP et MySQL :

## (création de base et de table)

```
$requete="CREATE TABLE PROF
(
 ID INT AUTO_INCREMENT NOT NULL,
 NOM VARCHAR (50) NOT NULL,
 PRENOM VARCHAR (50),
 TARDES VARCHAR (7),
 PRIMARY KEY (ID)
);";
$base = mysql_connect(localhost,"user",""); // connexion à MySQL
$id_db = mysql_create_db("SERECOM") // Création de la base SERECOM
 or die("Probleme lors de la création de la base SERECOM
");
mysql_select_db("SERECOM",$base) // Sélection de la base SERECOM
if (mysql_query($requete, $base)) // envoie d'une requête MySQL à la base $base
{ echo "table créée
"; }
else
{ echo "Pb with mysql_query
"; }
```

# PHP et MySQL :

## (insertion et suppression d'éléments)

```
// insertion d'elements
```

```
$reqInsert = "INSERT INTO PROF (NOM, PRENOM, TARDES)
VALUES (\\"winckler\\", \\"marco\\", \\"-XXXXX-\\");";
mysql_query($reqInsert, $base);
```

```
$newName = "winckler";
```

```
$reqInsert = "INSERT INTO PROF (NOM, PRENOM, TARDES)
VALUES (\\"$newName\\", \\"marco\\", \\"-XXXXX-\\");";
mysql_query($reqInsert, $base);
```

```
// suppression d'elements
```

```
$newName = "winckler";
$reqInsert = "DELETE FROM PROF where nom=\\"winckler\\"";
mysql_query($reqInsert, $base);
```

```
$newName = "winckler";
```

```
$reqInsert = "DELETE FROM PROF where nom=\\"$newName\\"";
mysql_query($reqInsert, $base);
```

```
$reqInsert = "DELETE FROM prof";
mysql_query($reqInsert, $base);
```



# PHP et MySQL : récupération des résultats d'une requête de sélection

- SELECT retourne des lignes de résultats

```
SQL > SELECT * FROM users;
```

| ID | NAME | ADDRESS |
|----|------|---------|
|----|------|---------|

|   |         |       |
|---|---------|-------|
| 1 | Jacques | Paris |
|---|---------|-------|

← 1ère ligne

|   |        |            |
|---|--------|------------|
| 2 | George | Washington |
|---|--------|------------|

← 2ème ligne

- Les fonctions mysql clés
  - `mysql_fetch_row($result)` : retourne une ligne de résultat sous la forme d'un **tableau**. Les éléments du tableau étant les valeurs des attributs de la ligne. Retourne FALSE s'il n'y a plus aucune ligne. Accès aux valeurs par les indices 0,1,2, ...
  - `mysql_fetch_array($result, $option)` : retourne un **tableau associatif**. Les clés étant les noms des attributs et leurs valeurs associées leurs valeurs respectives. Retourne FALSE s'il n'y a plus aucune ligne.
    - *\$option* = MYSQL\_ASSOC, MYSQL\_NUM, and MYSQL\_BOTH (*par défaut*).

# PHP et MySQL : exemple

## (récupération des résultats d'une requête)

```
$reqSelect = "SELECT * FROM PROF
 WHERE NOM=\"winckler\" AND PRENOM=\"marco\"";
$resSelect = mysql_query($reqSelect, $base);
```

```
while ($ligne = mysql_fetch_row($resSelect))
{
 for ($i=0; $i<count($ligne); $i++) { echo "-- $ligne[$i] "; }
 echo "
";
}
```

OU ALORS :

```
while ($ligne = mysql_fetch_array($resSelect))
{
 foreach ($ligne as $champs=>$valeur) { echo "$champs = $valeur"; }
 echo "
";
}
```

# PHP et MySQL : exemple (combinaison requête et formulaire)

```
<HTML><BODY>
<?php
if ($QUERY_STRING != "") {
 $base = mysql_connect(localhost,"user",""); // connect. à la DB
 // selection de la base SERECOM
 mysql_select_db("SERECOM",$base)
 or die ("pb à la selection de SERECOM
");

 // construction de la requete SQL
 $listeStrParam=split("&",$QUERY_STRING);
 foreach ($listeStrParam as $strParam) {
 $param = split("=", $strParam);
 switch ($param[0]) {
 case "champsSelect": $champsSelect=$param[1];break;
 default : echo "parametre imprévu
";
 }
 }
 $requete="select $champsSelect from PROF;";
 $resSelect = mysql_query($requete, $base);
 if ($resSelect!="") {
 while ($ligne = mysql_fetch_array($resSelect, MYSQL_ASSOC))
 {
 foreach ($ligne as $ch=>$val) {
 $strResult = $strResult . "$ch:$val ";
 }
 $strResult = $strResult . "\n";
 }
 } else { $strResult="aucun élément trouvé
"; }
```

```
echo "<textarea rows=$nbelt cols=100>";
echo "$strResult";
echo "</textarea>";
} // IF QUERY_STRING ...
?>

<FORM name=formulr method=get
 action=MySQLLandForm.php>
<H1>Composez votre requete :</H1>
select
<select name=champsSelect size=1>
 <option value=*>*</option>
 <option value=nom>nom</option>
 <option value=prenom>prenom</option>
</select>
from SERECOM

<input type=submit value="clie here">
</FORM></BODY></HTML>
```

# Compléments

- Web :
  - [www.php.net/manual/en/](http://www.php.net/manual/en/)
- Livres :
  - PHP, collection le tout en poche, M. Dreyfus, Campus Press (ed.)
  - Pratique de MySQL et PHP, P. Rigaud, O'Reilly (ed.)
  - PHP cookbook (UK), S. Hughes, SAMS (ed.)

# Manipulation de fichiers

- Mécanismes permettant la manipulation de fichiers coté serveur
  - `$fp = fopen($nom_file [, $mode])` : ouverture du fichier identifié par son nom `$nom_file` et dans un mode `$mode` particulier, retourne un identificateur `$fp` de fichier ou FALSE si échec.
    - r : autorise la lecture
    - r+ : autorise lecture et écriture en mode surfrappe
    - w : autorise l'écriture et efface ou créé le fichier
    - w+ : autorise lecture et écriture et efface ou créé le fichier
    - a : autorise l'écriture et ajoute en fin de fichier ou créé le fichier
    - a+ : autorise lecture et écriture et ajoute en fin de fichier ou créé le fichier
  - `fclose($fp)` : ferme le fichier identifié par le `$fp`

# Manipulation de fichiers

- `file_exists($nom_file)` : indique si le fichier `$nom_file` existe
- `filesize($nom_file)` : retourne la taille du fichier `$nom_file`
- `filetype($nom_file)` : retourne le type du fichier `$nom_file`
- `readfile($nom_file)` : affiche le fichier `$nom_file`
- `unlink($nom_file)` : supprime le fichier `$nom_file` du disque dur
- `copy($nom_source, $nom_dest)` : copie le fichier `$nom_source` vers `$nom_dest`
- `rename($nom_old, $nom_new)` : renomme le fichier `$nom_old` en `$nom_new`
  
- `fgets($fp[, $length])` : lit une ligne [de `$length` caractères au maximum]
- `fscanf ($fp, $format, $var1,$var2,...)` : lecture formatée d'un fichier
  - `$format = "%s * %i %f "` ... pour lire la ligne "nom \* 28 0.3"
  - Retourne un entier indiquant le nombre de variables ayant pu être initialisées
- `fputs ($fp, $str)` : écrit la chaîne `$str` dans le fichier identifié par `$fp`
  
- `rewind($fp)` : repositionne le pointeur au début du fichier
- `fgetc($fp)` : lit un caractère dans le fichier
- `feof($fp)` : *true* si fin de fichier

# Manipulation de fichiers

- Quelques fonctions pour manipuler les répertoires
  - `opendir($str)` : Ouvre le dossier `$str`, et récupère un pointeur `$d` dessus si succès, *FALSE* sinon
  - `closedir($d)` : Ferme le pointeur de dossier `$d`.
  - `getcwd()` : Retourne le nom du dossier courant (en format chaîne de caractères)
  - `dirname($str)` : Retourne le chemin d'accès au dossier courant
  - `basename($str)` : retourne le nom du fichier/répertoire sans le chemin d'accès.
  - `is_dir($str)` : *true* si c'est un répertoire (*false* si c'est un fichier)
  - `readdir($d)` : Lit **une** entrée du dossier identifié par `$d` i.e. retourne un nom de fichier/répertoire de la liste des fichiers/répertoires du dossier pointé. Les fichiers ne sont pas triés  
Retourne *FALSE* s'il n'y a plus de fichier
  - `rewinddir($d)` : repositionne le pointeur au début du répertoire
  - `chdir($str)` : Change le dossier courant en `$str`. Retourne *TRUE* si succès, sinon *FALSE*.
  - `rmdir($str)`, `mkdir($str)` : supprime ou crée un répertoire

# Manipulation de classes en PHP : exemples

```
class Voiture { // déclaration de la classe
 var $couleur; // déclaration d'un attribut
 var $belle = TRUE; // initialisation d'un attribut
 function voiture() { // constructeur
 $this->couleur = "noire";
 } // le mot clé $this faisant référence à l'objet est obligatoire dans la
 // déclaration de la classe
 function Set_Couleur($couleur) {
 $this->couleur = $couleur;
 }
}

$mavoiture = new Voiture(); // création d'une instance
$mavoiture->Set_Couleur("blanche"); // appel d'une méthode
$scoul = $mavoiture->couleur; // appel d'un attribut
```