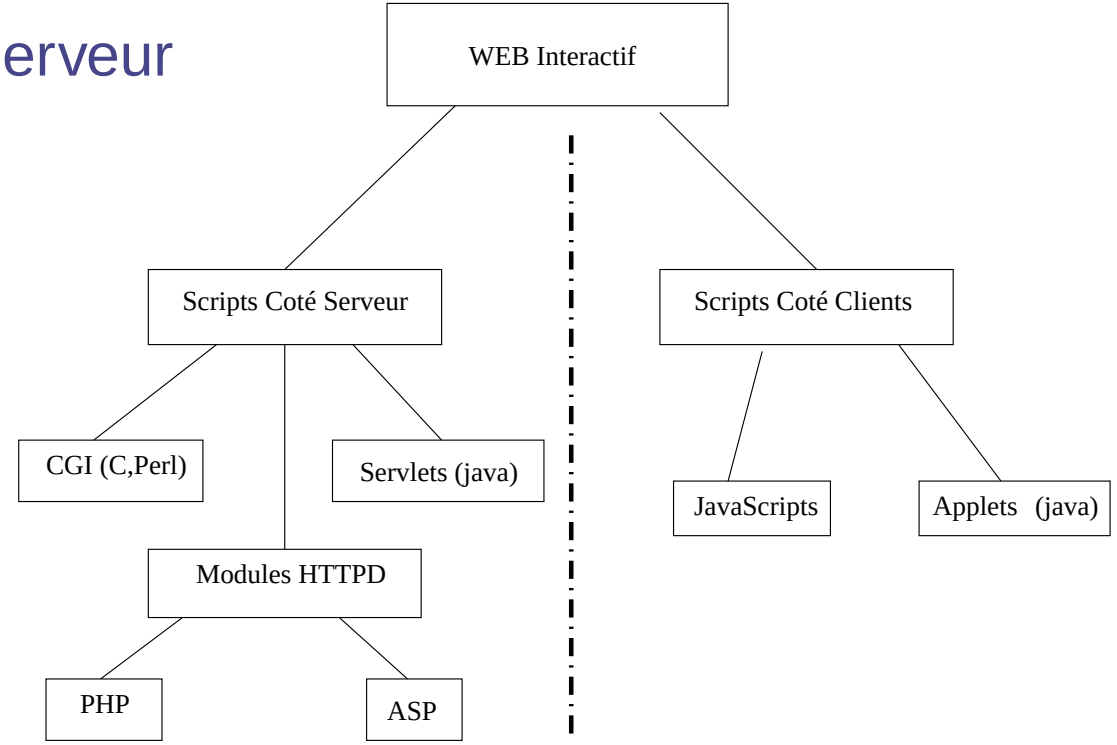
Introduction

Introduction

- Mise en relation des bases de données et des applications
- Création interfaces graphiques
- Utilisation et administration de l’application
- Base de données et WEB

Le Web Interactif

- Interaction entre le client et le serveur
- Pages dynamiques : dépendent des données fournies par le client
- Deux principaux types d'interactions
  - Coté client
  - Coté Serveur



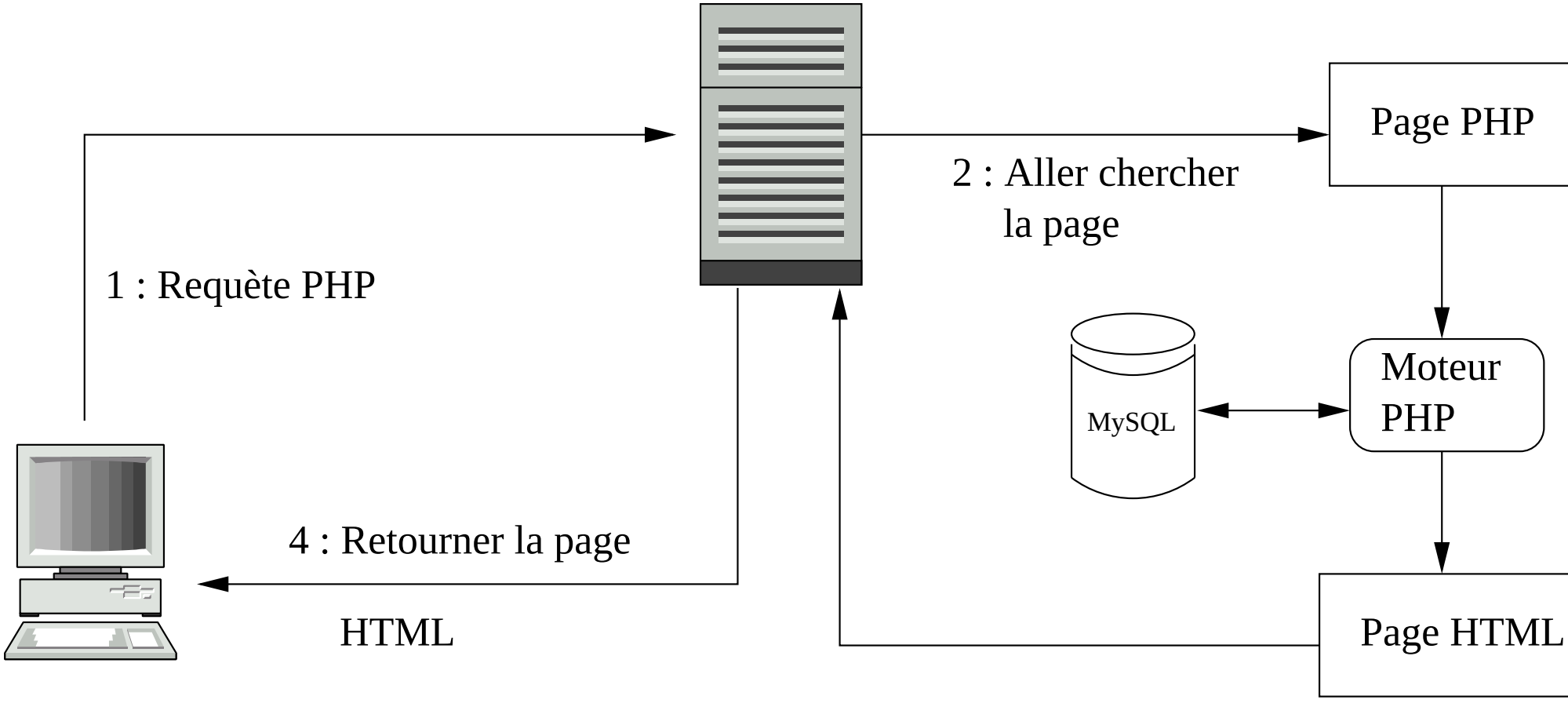
Vue d’ensemble de PHP

- Langage de scripts
- coté serveur
- Embarqué dans les pages HTML
- Syntaxe héritée du C et du PERL
- Extensible (nombreuses bibliothèques)
- open source - Licence GPL

Historique

- 1994 : Création par Ramsus Lerdorf pour des besoins personnels **Personal Home Page**
- 1995 : Première version publique
- 1997 : Devient un travail d'équipe sous la houlette de Zeev Suraski et Andi Gutmans
  - interpréteur réécrit (Version 3)
  - Renommé PHP : **PHP is HyperText Preprocessor**
- 2000 : Dernière version PHP4
  - Possibilité de programmation Objet
  - juillet 2000 : 300 000 sites sous php
- vers PHP 5 et + d'objets

Fonctionnement



Avantages

- Le client n’a pas accès au code source puisque celui ci est interprété
- Programmation simple
- Gratuit
- Nombreuses bibliothèques : Math, SSL, SGBD

Inconvénients

- Pas aussi rapide que certains scripts CGI
- Pas d’interactivité au niveau client
  - Celle ci se fera aux moyens des formulaires html

Liens

- `http://www.php.net`
- `http://www.phpinfo.net`
- `http://www.phpfrance.com`

Documentation en ligne

- http://fr.php.net/<function name>
- Voir exemple : echo

Le classique : Bonjour

```
<HTML>
<BODY>
<H1> Mon Premier Script </H1>
<?PHP
echo "HELLO WORLD";
?>
</BODY>
</HTML>
```

Remarque

- Même en local, on doit passer par le serveur WEB
- Sinon petit problème (dépend du client WEB !)

Un autre exemple

```
<HTML>
<BODY>
<H1> Mon deuxième Script </H1>
<?PHP
echo "Nous sommes le " . gmdate("D j M Y") . "<br>";
echo "Je rajoute deux sauts à la ligne <br><br>";
echo "et <u>un texte souligné</u>";
?>
<H1>Continuons</H1>
Du html de base
</BODY>
</HTML>
```

Base de la Syntaxe

Intégration dans une page HTML

- Les pages WEB dynamiques générées avec PHP4 sont au format php
- Le code source PHP est à insérer dans le code HTML avec les balises
  - **<?php** – Balise de début
  - **?>** – Balise de fin
- Chaque instructions du script PHP se termine par **;**

Les commentaires

- Les commentaires PHP se font comme en C
  - // Ceci est un commentaire de fin de ligne
  - # Ou comme cela
  - /* Ceci est un autre commentaire */
- Tout ce qui se trouve en commentaires est ignoré
- Commenter un code PHP est essentiel : très difficile à maintenir

Affichage

- 3 fonctions d’affichage sur le navigateur
  - echo : le plus souvent utilisé
  - print
  - printf : a la C

Types de données

- PHP supporte les types de données suivantes :
  - booléen
  - Nombres entiers
  - Nombre à virgule flottante
  - Chaînes de caractères
  - tableaux
  - Objets

Variables

- Tous les noms de variables sont précédés d'un $
- Le typage des variables est implicite en PHP
- Il n'est pas nécessaire de les déclarer avant utilisation **!!!**

Exemple

- $toto = 12;
- $titi = "bonjour";

Variables et fonctions

- **isset($var)** : renvoie vrai si la variable existe
- **empty($var)** : renvoie vrai si la variable est vide
- **unset($var)** : détruit la variable
- **gettype($var)** : retourne le type de la variable
- **islong($var)**, **isdouble($var)** ... : renvoie vrai si la variable est de type associé

Exemple

```
<HTML>
<BODY>
<H1> Mon troisième Script </H1>
<?PHP
$var = 10;
echo "Existe ?" . isset($var) . "<br><br>";
echo "Vide ? " . empty($var) . "<br><br>";
echo "Le type ? " .gettype($var) . "<br><br>";
$var = 'toto';
echo "Le type ? " .gettype($var) . "<br><br>";
unset($var);
echo "Existe ?" . isset($var) . "<br><br>";
?>
</BODY>
</HTML>
```

Une petite spécificité

- Une variable peut avoir pour identificateur la valeur d'une autre variable
- Syntaxe : ${$var}
- Complique le déboguage !!! : utiliser avec parcimonie

Exemple

```
$toto = "foobar";
${$toto} = 2002;
echo $foobar;              // Affiche la valeur de $foobar = 2002
```

Booléens

- Les variables booléennes prennent comme valeur TRUE et FALSE
- Une valeur entière nulle est considérée comme false
- Une chaine de caractère vide aussi
- Une chaîne de caractères : "0" aussi **!!!**

Chaînes de caractères

- une chaîne de caractères n'est pas limitée en taille
- Elle peut être définie de la façon suivante
  - $toto = 'Bonjour';
  - $toto = "Bonjour ";
- Les doubles quotes permettent l'évaluation des variables et caractères spéciaux
  - echo 'Alors : $toto'; **// affiche Alors : $toto**
  - echo "Alors : $toto"; **// affiche Alors : Bonjour**
- Quelques caractères spéciaux : $\n$ nouvelle ligne, $\t$ tabulation, \$ dollar...

Fonctions et Chaînes de caractères

- Opérateur de concaténation : .
- **strlen($var)** : la taille
- **strtolower($var)** : convertit en minuscules
- **strtoupper($var)** : convertit en majuscules
- **trim($var)** : suppression des espaces de début et de fin
- **strnatcmp($var,$var2)** : comparaison de 2 chaînes

- **HtmlSpecialChars($var)** : Convertir des balises HTML en châines

Exemple

```
<HTML>
<BODY>
<H1> Utilisateurs malveillants !! </H1>
<?PHP
$ch = "Salut ";
$ch2 ="<u>Tout le monde</u>";
$ch3 = $ch . $ch2;
echo $ch3 . "<br><br>";
echo HtmlSpecialChars($ch3) ?>
</BODY>
</HTML>
```

Les constantes

- Le programmeur peut définir des constantes dont la valeur est connue une fois pour toute
- Les constantes ne portent pas le $
- Contrairement aux variables, les identificateurs de contantes ne sont pas sensibles à la casse
- **define("CTE",VALEUR)**
- Exemple
  - define("YEAR",1999)
  - define("Ville","Lens")
- Convention : CONSTANTES en majuscules

Les tableaux

- Une variable tableau est de type **array**
- Les éléments du tableau peuvent être de type différent **!!**
- La taille du tableau peut varier en cours de route
- Comme en C, le premier élément a pour index 0
- On peut initialiser un tableau

```
$tabcolors = array('red','green','blue');
```

- Mais il peut être initialisé également au fur et à mesure

```
$prenoms[] = 'Gilles';
$prenoms[] = Stéphanie';
```

Parcours d'un tableau

- Première solution

```
$i = 0
while(i<count($tab))
        echo $tab[$i++] . "<br>";
```

- Deuxième solution

```
foreach($tab as $elem)
        echo $elem
```

Tableaux et Fonctions

- **count($tab)** retourne
  - La taille du tableau s'il existe
  - 1 si la varable $tab n'est pas un tableau
  - 0 si la variable n'existe pas
- **in_array($var,$tab)** : retourne true si la valeur $var existe dans $tab
- **list($var1,$var2...)** : transforme une liste de variables en talbeau
- **shuffle($tab)** : mélange le contenu du tableau
- **sort($tab)** : tri suivant l'ordre alphanumérique du tableau
- **rsort($tab)** : tri suivant l'ordre alphanumérique inverse

Tableaux ... suite

- Il est possible d'effectuer des opérations complexes sur les tableaux
- Propre opération de comparaison des éléments : Tri personnel
- **Attention** : Les variables tableaux ne sont pas évaluées lorsqu'elles sont au milieu d'une chaîne de carctères !

```
echo "$tab[3]"; //Invalide
echo $tab[3]; //OK
```

Tableaux associatifs

- On associe à chacun des éléments une clé : HashTable
- L'initialisation est similaire à celle d'un tableau normal

Exemple 1

```
$personne = array("Nom" => "Audemard","Prenom" =>
"Gilles");
```

Exemple 2

```
$personne["Nom"] = "Audemard";
$personne["Prenom"] = "Gilles";
```

"Audemard" est associée à la clé "Nom" ....

Parcours d’un tableau associatif

- Première solution

```
foreach($Personne as $elem)
        echo $elem
```

- Deuxième solution

```
foreach($Personne as $key => $elem)
        echo "$key associée avec $elem"
```

Tableau associatif ... le retour

- Chaque tableau associatif entretient un pointeur courant qui sert à naviguer grace à certaines fonctions
  - **reset($Personne)** : place le pointeur sur le premier élément et retourne sa valeur
  - **next($Personne)** : déplace le pointeur sur le suivant et retourne sa valeur
  - **prev($Personne)** : déplace le pointeur sur le précédent et retourne sa valeur
  - **current($Personne)** : retourne la valeur courante du pointeur
  - **each($Personne)** : retourne la paire clé/valeur courante

Les opérateurs

- Arithmétiques : + (addition), - (soustraction), *(multiplication), / (division) % (modulo)
- Bits à bits : & (et), | (ou), ^(xor), ~ (non), << et >> (décalage)
- Logiques : && and (et), || or (ou), xor, ! (négation)
- Affectation : (=) peut être combinée avec d'autres opérateurs
- [in][de]crementation : ++ et –
- Comparaison : == (égalité) ....
- ternaire : (cond ? ... : ...)

Instructions de contrôles

- Blocs d'instructions délimités par { et }
  - Portée des variables locales....
- Même syntaxe que le langage C
- Structures conditionnelles : if, else, switch, elseif
- Structures de boucles : for, while, do...while, foreach
- Les sauts : break et continue

Arrêt prématuré

- On peut stopper prématurement un script
  - **die** : arrète le script et affiche le contenu de la chaîne de caractères passée en paramètres
  - **exit** : arrète le script mais sans afficher de message d'erreur
- Ces fonctions stoppent tout le script, pas seulement le bloc en cours

Inclusions

- On peut inclure dans un script PHP, le contenu d'un autre fichier
  - **require** : insère dans le code le contenu du fichier passé en paramètre, même si celui ci n'est pas du PHP
  - **include** : évalue et insère à chaque appel (même dans une boucle) le contenu du fichier passé en argument. On peut donc l'utiliser facilement avec une variable passée en paramètre

Variables prédéfinies

- HTTP_USER_AGENT : signature du navigateur du client
- REMOTE_ADDR : adresse IP du client
- QUERY_STRING : chaîne au format url des paramètres de la page en cours

Constantes prédéfinies

- __FILE__ : nom du fichier en cours
- __LINE__ : numéro de ligne en cours
- TRUE
- FALSE
- PHP_VERSION : version de PHP utilisée
- PHP_OS : système d'exploitation du serveur

Les fonctions

Les fonctions

- Les fonctions peuvent prendre des arguments dont il n’est pas besoin de spécifier le type
- Elles peuvent retourner une valeur
- tout type peut être renvoyé : tableau, entier ...
- Les identificateurs de fonctions sont insensibles à la casse
- En PHP3 une fonction doit être définie avant d’être utilisée
- Cela n’est plus vrai en PHP4

Corps d'une fonction

```
function <name>($arg1,$arg2,$arg3....) {

        # Corps de la fonction

        return valeur_de_retour                    # Optionnel
}
```

Arguments de fonctions

- Les arguments sont séparés par une virgule
- Les arguments peuvent être passés
  - par valeur (défaut) : Les variables ne sont pas affectées par les changements au sein de la fonction
  - par référence : Les variables sont affectées par les changements au sein de la fonction
    - De façon permanente : en ajoutant un & devant le nom du paramètre
    - De façon temporaire : en ajoutant un & devant le nom de la variable lors de l'appel de la fonction !!

Arguments de fonctions

- enfin :
  - par défaut d'arguments :
    - il suffit de spécifier la valeur par défaut avec un = dans la déclaration de la fonction
    - Cette valeur doit être une constante
    - Les arguments par défaut doivent être les derniers de la liste

Voir exemple

Retourner plusieurs valeurs

- Grace à une astuce, il est possible de retourner plusieurs valeurs !!
- Il suffit de retourner un tableau et d'utiliser la fonction **list**

```
function trigo($angle) {
        return array(sin($angle),cos($angle),tan($angle);
}
list($a,$b,$c) = trigo(0)
echo "sin(0) = $a, cos(0) = $a, tan(0) = $a";
```

Un peu d’objets

Introduction

- Tous les attributs et méthodes sont publics
- Le mot clé **this** faisant référence à l'instance courante est obligatoire
- Existence de constructeurs
- pas de destructeur : Garbage Collector
- Tout objet instancié est une variable et peut donc être passé en argument de fonctions ou être un retour de valeur de fonction
- Héritage simple uniquement (mot clé **extends**
  - Possibilité de surcharge de fonctions
  - Possibilité de redéfinition de fonctions
  - Possibilité de création de nouvelles fonctions

Exemple

```
class Caddie {
        var $elements;                                          // Elements de notre caddie

        function Caddie() {                                     // Constructeur : on met un cadeau
                $this→add("cadeaux",1);
        }

        function add($type,$nbre) {                             // On ajoute au caddie
                $tmp = array($type=>$nbre);
                $this→$elements = array_merge($this→$elements,$tmp);
        }

        function del($type,$nbre) {                             // On enlève au caddie
                if($this→$elements[$type] >=$nbre) {
                        $this→$elements[$type] -=$nbre;
                        return TRUE;
                } else return FALSE;
        }

}
```

Utilisation

```
$monCaddie = new Caddie();
$monCaddie→add("imprimante",1);
$monCaddie→add("scanner",1);
```

Quelques fonctions

- **get_declared_classes()** : retourne un tableau listant toutes les classes définies
- **class_exists($str)** : renvoie vrai si la classe de nom $str existe
- **get_class($obj)** : retourne le nom de la classe de l'objet $obj
- **get_class_methods($cl)** : retourne un tableau listant toutes les méthodes de la classe $obj
- **method_exists($cl,$m** : renvoie vrai si $m est une méthode de la classe $cl
- **....**

Passage de paramètres à un script

Introduction

- Qui dit page web dynamique, dit possibilité au client de choisir la valeur de certaines variables
- Le langage HTML propose l'utilisation de formulaires afin de collecter diverses informations
- Ces informations seront ensuite envoyés à d'autres pages HTML ou autres PHP

Syntaxe

- les balises <**FORM**> et <**/FORM**> servent à délimiter les formulaires
- Les paramètres pris par le tag <**FORM**> sont
  - ACTION : fixe le nom du document qui sera appelé lors de la soumission du formulaire
  - METHOD : détermine la méthode de passage des paramètres lors de la soumission des données
    - GET
    - POST

Les Méthodes de passage

- Méthode GET
  - Les paramètres (et les valeurs) sont passés dans l'URL du document de la façon suivante :

    document.php?param1=val1&param2=val2...
  - Une adresse URL étant limitée en taille, certaines fois on ne peut pas passer tous les paramètres
- Méthode POST
  - Les paramètres sont passés sous le même format mais ils sont passés dans le corps de la requête HTTP
  - Ils sont invisibles
    - sur la ligne de commande
    - sur le log du serveur web

Quelles méthodes choisir ?

- Dépend de ce que l'on veut faire
- La méthode GET est inadaptée à la saisie de mots de passe
- La méthode POST impose obligatoirement une reconnexion au serveur
  - C'est pourquoi, les moteurs de recherche choisissent la méthode GET
  - Sur peu de jours, la base de donnée change peu d'état, deux mêmes recherches aboutissent à peu près au même résultat

Les éléments du formulaire

- Eléments utilisant le paramètre **INPUT**
  - Ligne de saisie de texte
  - Zone d'acquisition de mot de passe
  - Bouton (validation, reset)
  - Case à cocher, Bouton radio
- Zone de de texte
- Menu déroulant

Ligne de saisie de texte

- paramètre possibles
  - TYPE : text
  - VALUE : valeur par défaut
  - NAME : la variable associée à la zone de texte saisie

```
<FORM>
    Titre Chanson :
    <INPUT TYPE="text" VALUE="Ma chanson préférée"
    NAME="champs" SIZE="40">
</FORM>
```

- Voir exemple

Saisie de mot de passe

- paramètre possibles
  - TYPE : password
  - VALUE : valeur par défaut
  - NAME : la variable associée au mot de passe

```
<FORM>
    Password : <INPUT TYPE="password" VALUE="toto"
    NAME="champs" SIZE="20">
</FORM>
```

- Attention : PAS de méthodes GET dans ce cas

Les boutons

- Paramètres possibles
  - Type
    - Submit (soumission du formulaire)
    - reset (init de la valeur initiale)
  - VALUE : Le texte du bouton

```
<INPUT TYPE="submit" VALUE="OK">
```

Les cases à cocher

- Paramètres possibles
  - type : checkbox
  - CHECKED : coché initialement
  - VALUE : la valeur prise par la variable si coché

```
<FORM>
    La bouillabaisse
    <INPUT TYPE="checkbox" NAME="champs[0]"
    CHECKED VALUE ="Bouillabaisse">

    La carbonnade
    <INPUT TYPE="checkbox" NAME="champs[1]"
    VALUE ="Carbonnade">
</FORM>
```

Les boutons radios

- Paramètres possibles
  - type : radio
  - CHECKED : coché initialement
  - VALUE : la valeur prise par la variable si coché
- 1 seul bouton peut être coché

```
La bouillabaisse
<INPUT TYPE="radio" NAME="champs" CHECKED
VALUE="Bouillabaisse">

La carbonnade
<INPUT TYPE="radio" NAME="champs"
VALUE="Carbonnade">
```

Le menu déroulant

- Introduit par les tags <**SELECT**>
- Puis chaque valeur possible est précédée d'un tag <**OPTION**>
- La valeur par défaut possède le paramètre **SELECTED**

```
<FORM>
    <SELECT NAME="champs">
        <OPTION SELECTED> Bouillabaisse
        <OPTION > Carbonnade
    </SELECT>
</FORM>
```

Les sessions

Introduction

- Comment garder la valeur des variables tout au long des pages PHP
- Jusqu'à la version 3
  - Passage des variables dans l'url
  - Passage caché via la méthode POST et le tag HIDDEN
  - Utilisation d'une base de donnée
  - Utilisation de cookies
- Depuis la version 4 : **Les sessions**

Introduction ... 2

- Moyen de sauvegarder et de modifier des variables tout le long de la visite d'un internaute
- Sécurisation de sites, espionnage :-(, sauvegarde du panier
- Les informations relatives à une session sont conservées sur le serveur
- Un identifiant de session est posté sur le client via un cookie ou via l'url

Comment ça marche

- Une session doit obligatoirement démarrer avant l'envoi de toute information chez le client (affichage, ou envoi de header)
- Une session démarre avec **session_start()**

```
<?php
session_start();                                  // Une nouvelle session
echo "<HTML><BODY>";
?>
<form action = 'menu.php?<?=SID?>' method='post'>
<input type='text' name='login'>
<input type='password' name='password'>
<input type='submit' value='entrer'>
</form>
```

Comment ça marche ... 2

- La variable $PHPSESSID est le nom par défaut attribué à la session
- Donc sur les pages internes, on regarde si la personne est bien connectée

```
<?php
if($PHPSESSID) {
        session_start($PHPSESSID);
        } else {
        echo"session expirée ou invalide";
        echo"<a href='index.php'>reconnexion</a>"
        exit();
        }
?>
```

Comment ça marche ...3

- Ensuite, on peut enregistrer des variables de sessions
- La fonction **session_register()** s'occupe de cette tâche
- La fonction **session_unregister()** permet de supprimer des variables de session

```
<?php
// Récupérer le login dans la variable vlog
// Enregistrer vlog dans la session
session_register("vlog");
....
?>
```

Comment ça marche ... 4

- Comme nous l'avons dit, on peut faire passer l'identifiant de session
  - par cookie
  - par l'url
- Si le client n'accepte pas les cookies, la première technique est vouée à l'échec
- Mieux vaut utiliser la deuxième
- Il faut alors faire comme ceci

```
<a href='page.php?<?=SID?>'>cliquer ici</a>
```

Comment ça marche ... 5

- Dans le cas du passage par paramètre, l'écoute du réseau permet d'obtenir la clé de session
- Et de se connecter ainsi en se faisant passer pour un tiers
- Une astuce pour éviter ceci
- Enregistrer au début de la session l'url du client et la comparer à chaque nouvelle requête http.
- Elle ne marche pas si le pirate à la même url !

```
<?php
session_start(); // Une nouvelle session
$ip = $REMOTE_ADDR;
session_register("ip");
?>
```

PHP et Base de données

Introduction

- PHP permet une intégration asiée avec des SGBDR
- En premier lieu, citons MySQL
- Mais, cela ne s’arrête pas là
- En effet, On peut administrer des bases de données Oracle, ODBC (Access), PostGresQL

Introduction ...2

- Chaque SGBD possède sa propre API
- Mais les mécanismes de consultation sont globalement les mêmes pour tous
- Ce n'est pas comme JDBC, Quand on change de base de données, on doit changer le nom des fonctions !!!
- il existe de packages pour éviter cela (adodb)
- Etude avec les fonctions MySQL

Mise en place de la connexion

- Fonction **mysql_connect("server","username","passwd");**
- Retourne
  - un index de connexion (entier) si la connexion à réussi
  - FALSE sinon

- Sélection de la base : **mysql_select_db("base");**
- Restourne TRUE en cas de succès, FALSE sinon

Connexion persistante

- Dans le cas précédent, la connexion sera fermée à la fin du script PHP
- On peut vouloir ne pas refaire la connexion à chaque nouveau script
- Création d'une connexion persistante
- Fonction **mysql_pconnect("server","username","passwd");**
- même comportement que pour la fonction mysql_connect
- Après un certain temps d'inactivité, la connexion est automatiquement fermée

Fermeture de la connexion

- Fonction **mysql_close($id)**
- Simple, mais nécessaire

Exemple de connexion

```
<?php
if( $id = mysql_connect( localhost , foobar , 0478 ) ) {
        if( $id_db = mysql_select_db( gigabase ) ) {
                echo "Succès de connexion." ;
                /* code du script */
        } else {
                die( "Echec de connexion à la base." );
        }
        mysql_close($id);
} else {
        die( "Echec de connexion au serveur de base de données." );
        }
}
?>
```

Exécution d'une requête

- La fonction **mysql_query($query)** s'occupe de cette tâche
- **$query** est la requête à exécuter

```
$result = mysql_query("Select nom,age From Table Where ...")
```

- Diverses fonctions permettent d'extraire les données du résultat de la requête

Récupération des données

- Les données se récupère ligne par ligne
- Fonction **mysql_fetch_row($result)**
  - retourne une ligne du résultat sous la forme d'un tableau
  - retourne FALSE s'il n'y a plus aucune ligne

```
if($result = mysql_query("Select nom,age From Table Where ...")) {
        while($ligne=mysql_fetch_row($result)) {
                echo $ligne[0]." a ". $ligne[1] . "ans<br>";
        }
else {
        echo "Problème requête";
}
```

Récupèration des données ... 2

- il est également possible de récupérer les données dans un tableau associatif
- Fonction **mysql_fetch_array($result)**

```
if($result = mysql_query("Select nom,age From Table Where ...")) {
        while($ligne=mysql_fetch_array($result)) {
                echo "$ligne["nom"] a $ligne["age"] ans<br>";
        }
else {
        echo "Problème requête";
}
```

Récupèration des données ... 3

- Enfin, il est également possible de récupérer les données dans un objet
- Fonction **mysql_fetch_object()**

```
if($result = mysql_query("Select nom,age From Table Where ...")) {
        while($ligne=mysql_fetch_object($result)) {
                echo "$ligne->$nom a $ligne->$age ans<br>";
        }
else {
        echo "Problème requête";
}
```

Quelques fonctions

- **mysql_num_fields($result)** retourne le nombre d'attributs de la requête
- **mysql_num_rows($result)** retourne le nombre de lignes de la reqûete
- **mysql_list_tables()** retourne le nom des tables de la base de données
- **mysql_list_fields("bd","table")** retourne le nom des champs de la table dans la base de donnée bd

Modification des données

- La modification se fait aussi avec la fonction **mysql_query**
- La fonction **mysql_affected_rows()** retourne
  - le nombres de lignes affectées par la modification
  - -1 si la dernière requête a échoué
- Il faut l'utiliser juste après la modification

Traitement des erreurs

- La fonction **mysql_error()** retourne le dernier message d'erreur mySQL
- La fonction **mysql_errno()** retourne le numéro de la dernière erreur

Quelques conseils

- Un code PHP peut rapidement ressembler à un véritable foutoir
- Ne pas lésigner sur les commentaires
- Segmenter le code
  - Aspect graphique
  - contenu
  - mise en forme
- Gérer de nombreux fichiers (entete, connexion à la base, pied de pages)

Encore

- Programmer objet (Penser à la version 5)
- Toujours contrôler les données saisies pour éviter tout problème de piratage

Aujourd'hui

- phpMyAdmin
- un exemple de site sous php : sondage
- Le TP a réaliser

Bonne Année

Passage de paramètres

- En tp, nous avons facilement transmit des variables d'une page à une autre
- Avant la version 4.2 de PHP, on procédait de cette manière
- $\rightarrow$ Problème de sécurité
- On peut désactiver cette façon de faire dans le script d'initialisation de php : php.ini
- Depuis la version 4.2 la variable `register_globals` est à OFF

Comment faire alors

- On va utiliser des tableaux (nécessite la variable `track_vars` à ON)
- Il existe un tableau associatif pour chaque type de passage de variable

$HTTP_GET_VARS	Permet de récupérer les variables passées par la méthode GET
$HTTP_POST_VARS	Permet de récupérer les variables envoyées par la méthode POST
$HTTP_COOKIE_VARS	Permet de récupérer les cookies
$HTTP_SESSION_VARS	Permet de récupérer les variables de session
$HTTP_ENV_VARS	Permet de récupérer les variables d'environnement données par PH
$HTTP_SERVER_VARS	Permet de récupérer les variables serveur

Depuis la version 4.1

- Depuis la version 4.1 le nom de ces tableaux a changé (trop long)

$HTTP_GET_VARS	$_GET
$HTTP_POST_VARS	$_POST
$HTTP_COOKIE_VARS	$_COOKIE
$HTTP_SESSION_VARS	$_SESSION
$HTTP_ENV_VARS	$_ENV
$HTTP_SERVER_VARS	$_SERVER

La fonction extract

- Une manière pour revenir à du plus classique

Supposons :
$_POST['nom']
$_POST['prenom']
$_POST['age']

extract($_POST,EXTR_OVERWRITE) crée les variables $nom, $prenom, $age

- Attention aux options....

Les options d'extract

EXTR_OVERWRITE	Ecrase les variables existantes
EXTR_SKIP	N'écrase pas les variables existantes
EXTR_PREFIX_SAME	Si une variable existe déjà,une nouvelle variable est créée avec un préfi x donné en 3ème argument à la fonction
EXTR_PREFIX_ALL	Crée de nouvelles variables avec le préfi x passé en 3ème argument pour toutes les clés du tableau
EXTR_PREFIX_INVALID	Crée de nouvelles variables avec le préfi x passé en 3ème argument pour les noms de variable invalides (par exemple $1)

Verifier l’existence d’un lien

- Fonction permettant de vérifier l’existence d’un lien

```
<?php
function existLink($site) {
        $site = "http://www.phpscripts-fr.net";
        $file = @fopen($site, ’r’);
        return $file
}
?>
```

des exemples

- résultat d’une requète dans un tableau
- choix d’une liste déroulante à partir d’une requète

Un exemple

- Exemple de sondage