

# PHP

## (Hypertext Pre-Processor)

Qu'est ce que PHP ?

Un langage de scripts généraliste (perl ?)

Pour nous : un langage interprété par Apache pour générer une page HTML standard qui est transmise au client.

Il s'agit alors de script exécuté « **coté serveur** ».

Pour nous : php4

Similaire : ASP (Active Server Pages), de chez Microsoft.

Syntaxe proche langage C.



# PHP

## (Hypertext Pre-Processor)

Qu'est ce que PHP ?

Un langage de scripts généraliste (perl ?)

Pour nous : un langage interprété par Apache pour générer une page HTML standard qui est transmise au client.

Il s'agit alors de script exécuté « **coté serveur** ».

Pour nous : php4

Similaire : ASP (Active Server Pages), de chez Microsoft.

Syntaxe proche langage C.

# PHP

## (Hypertext Pre-Processor)

- Différence avec des scripts « cgi » :
  - Script cgi : exécutable lancé par apache pour générer du HTML.
  - Php : code html dans lequel on insère des directives php. (gain de temps...)
- Coté client : réception du résultat, sans connaissance du code (sécurité)

# PHP fonctionnement

Ecrire du PHP :

- Un fichier php porte l'extension .php
- C'est un fichier html classique contenant des balises pour indiquer quelles parties sont « en php »
- On passe du « mode texte » au « mode php » avec la balise `<?php ?>`

```
<?php  
echo "bonjour";  
echo "le monde";  
?>
```

# PHP instructions

Php est très semblable au langage C :

- Instructions séparées par des virgules;
- Commentaires par // et /\* ... \*/
- Beaucoup de mots clefs communs (if, for, while, return, switch, == )

# PHP mixé à HTML

Exemple de changement de mode malin :

```
<?php  
if ($valeur = TRUE)  
{  
  ?>  
  <h1> C'est vrai </h1>  
  <?php } else { ?>  
  <h1> C'est faux </h1>  
  <?php } ?>
```

Surtout malin dans le cas de longs passages de texte html à écrire : évite les echo pénibles.

# Variables PHP

- Déclaration explicite du type d'une variable non nécessaire.  
(type d'une variable déterminé par le contexte d'utilisation.)
- Forcer un type :
- Transtypage (cast) :
- Obtenir le type d'une variable :
- Vérifier le type d'une variable :
- Noms de variables sensibles à la casse.

```
int settype ( string var , string type)
```

```
$mon_double = (double) mon_int;
```

```
echo gettype($ma_variable);
```

```
if is_int($ma_variable) {}
```

# Variables PHP

- Types scalaires

- Booléen
- Entier
- Nombre à virgule flottante
- Chaîne de caractères

```
$mon_bool=true;  
$mon_entier=0;  
$mon_float = 0.123;  
$ma_chaine = "on bitin";
```

- Types composés :

- Tableau
- Objet

```
$mon_tab = array(1,2,3);
```

Similaire : classes C++ (voir plus loin)

- Types spéciaux :

- Ressource
- Null

Choses bizarres (connexions ftp... voir plus loin)  
Absence de valeur. 

```
$var = NULL;
```

# Variab les PHP

Le relâchement sur le type des variables peut poser problème lors de comparaison : utiliser l'opérateur d'égalité

« === »

# Chaines de char PHP

- Spécifications de chaînes :

- Guillemets simples : nom de variables non développés.

```
$chaine = 'toto $var \n';
```

- Guillemets doubles : noms de variables développés.

```
$chaine = "toto $var \n";
```

- Syntaxe HereDoc : comme guillemets doubles sans guillemets...

```
$chaine = <<<EOD bonjour,  
je m appelle "$nom"  
EOD;
```

# Tableaux PHP

Définition :

- Un tableau php est une association ordonnée : chaque valeur est associée à une clef.
  - La clef est un entier ou une chaîne
  - La valeur peut être de tout type.

Cette implémentation permet de faire des dictionnaires, tables de hachage, des piles...

# Tableaux PHP

## Création de tableaux : la fonction array

```
$tab = array(15);  
$tab = array(12); // Ecrase le tableau précédent  
$a1 = array("un","deux","trois","nous irons au bois");  
$a2 = array(1 => "un","deux","trois","nous irons au bois");  
$a3 = array(1 => "un",2 => "deux",3 => "trois",4 => "nous irons ...");
```

# Tableaux PHP

## Manipulation d'éléments de tableaux

```
$stab = array(15);  
$stab[]=12;      // On ajoute 12 à la fin du tableau (index 1)  
$stab[1]=24     // on remplace 12 par 24  
  
unset($arr[0]); // supprime l'élément de clef 0  
unset($arr);   // supprime le tableau
```

# Tableaux PHP

Afficher un tableau : mot clef « foreach »

```
<?php  
$tab = array('rouge','bleu','vert','jaune');  
  
foreach ( $tab as $couleur )  
{ echo "Aimez vous la couleur $couleur?\n"; }  
  
/* Affiche : Aimez vous la couleur rouge? Aimez vous la couleur bleu?  
Aimez vous la couleur vert? Aimez vous la couleur jaune? */  
?>
```

# Tableaux PHP

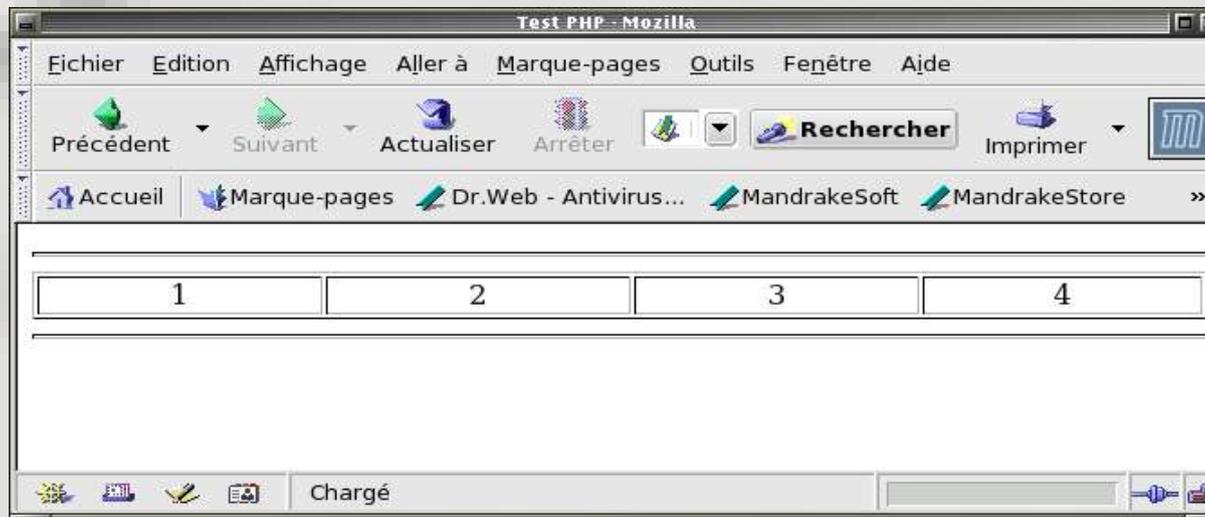
modifier tous les éléments d'un tableau :

```
<?php  
foreach( $stab as $cle => $couleur )  
{  
    // ne marche pas  
    //$couleur = strtoupper($couleur);  
    //marche :  
    $couleurs[$cle] = strtoupper($couleur);  
}  
?>
```

# Tableaux PHP

## Exercices :

- *Ecrire une page html qui affiche la page suivante :*



- *Ecrire une page php qui affiche un tableau 1D sous la forme précédente.*
- *Ecrire une page php qui fasse la même chose pour des tableau 2D.*

# Tableaux PHP

## Fonctions utiles de manipulations de tableaux :

- `reset($array)` Remet le pointeur interne de tableau au début.
- `pos($array)` Retourne la valeur de l'élément courant d'un tableau.
- `key($array)` Retourne l'indice de l'élément courant d'un tableau.
- `current($array)` Retourne la valeur de l'élément courant d'un tableau.
- `next($array)` Avance le pointeur interne d'un tableau.
- `prev($array)` Recule le pointeur courant de tableau.
- `end($array)` Positionne le pointeur de tableau en fin de tableau.
- `each($array)` Retourne la paire clé/valeur courante et avance le pointeur de tableau.

Chacune de ces fonctions renvoie false en cas de problème.

# AVANCEMENT

8 HEURES

# Tableaux PHP

Le mot-clef `list` : Transforme une liste de variables en tableau.

```
$info = array('rhum', 'transparent', 'alcool');
```

```
// Liste de toutes les variables
```

```
list ($drink, $color, $power) = $info;
```

```
echo "$drink est $color et l'$power le rend particulier.\n";
```

```
// Liste de certaines variables
```

```
list ($drink, , $power) = $info;
```

```
echo "$drink contient de l'$power.\n";
```

```
// Ou bien, n'utilisons que le troisième
```

```
list( , , $power) = $info;
```

```
echo "J'ai besoin d'$power!\n";
```

# Tableaux PHP

Utilisation de list : parcours de tableaux

```
$info = array('rhum', 'transparent', 'alcool');  
  
while (list ($key, $val) = each ($info))  
{  
    echo "$key => $val \n";  
}
```

# Tableaux PHP

## Fonctions et opérateurs utiles pour les tableaux :

- Opérateur + Ajoute deux tableaux. (clefs égales..)
- `array_merge($stab1, $stab2... )` Ajoute deux tableaux (clefs égales...)
- `count($stab)` Compte les éléments d'un tableau.
- `array_keys($stab)` Retourne toutes les clés d'un tableau
- `array_values($stab)` Retourne les valeurs d'un tableau
- `array_key_exists($key, $stab)` Regarde si une clef existe.
- `array_search($val,$stab)` Recherche la clé associée à une valeur.
- `array_flip($stab) :` Retourne un tableau : clefs <-> valeurs

Chacune de ces fonctions renvoie false en cas de problème.

# Tableaux PHP

Exemple de fusions de tableaux :

```
<?  
+ et merge...  
?>
```

# Tableaux PHP

Fonctions utiles de tris de tableaux :

- `sort ($array )` Trie un tableau par valeurs croissantes.
- `rsort ($array )` Trie un tableau par valeurs décroissantes.
- `ksort($array)` Trie un tableau par valeurs de clefs croissantes.
- `krsort($array)` Trie un tableau par valeurs de clefs décroissantes.
- `shuffle ($array )` Mélange les éléments d'un tableau.

Chacune de ces fonctions renvoie false en cas de problème.

# Pointeurs (?) PHP

- Assignment par référence

```
$a = 15;  
$une_ref=&$a;
```

Attention :

Ici, `une_ref` ne pointe pas sur `$a`,  
`$a` et `$une_ref` pointent sur 15

- Intérêt :
  - Modifications d'arguments par des fonctions...
  - Gain de temps pour copier des tableaux (mais une seule instance disponible)

# Noms de variables dynamiques

- Normal :

```
<?php $a = "bonjour"; ?>
```

- Variables variables :

```
<?php $$a = "monde"; ?>
```

- Une variable est créée de nom \$bonjour
- \$bonjour vaut "monde".
- \${\$a} vaut "monde".

# Noms de variables dynamiques

## Intérêt

```
$nom="a";  
$$nom=array(1,2,3,4);  
echo "<br> \$$nom ";  
affiche_tab1d($$nom);  
echo "<br>";
```

```
$nom="b";  
$$nom=array(11,22,33,44);  
echo "<br> \$$nom ";  
affiche_tab1d($$nom);  
echo "<br>";
```



# AVANCEMENT

10 HEURES

# Fonctions PHP

## Définir et appeler une fonction

```
function somme ($arg1, $arg2,$arg3)
{
    print "$arg1";
    $retval = $arg1+$arg2+$arg3;
    return $retval;
}
$toto=5;
$resu = somme(1,2,$toto);
print "$resu";
```

# Fonctions PHP

## Passage des arguments d'une fonction

Passage par valeurs : Une fonction ne modifie pas ses arguments.

```
function bidon ($arg1)  
{  
    $arg1=0;  
}  
  
$toto=15;  
bidon($toto);  
echo "toto = $toto";
```

# Fonctions PHP

## Passage de paramètres par référence

Pour pouvoir modifier une variable tout le temps

```
<?php  
function add_some_extra(&$string)  
{  
    $string .= ', et un peu plus.';  
}  
  
$str = 'Ceci est une chaîne';  
add_some_extra($str);  
echo $str; // affiche 'Ceci est une chaîne, et un peu plus.'  
?>
```

# Fonctions PHP

## Passage de paramètres par référence

Pour pouvoir modifier une variable ponctuellement

```
<?php  
function foo ($bar)  
{ $bar .= ', et un peu plus.'; }  
  
$str = 'Ceci est une chaîne';  
foo ($str);  
echo $str; // affiche 'Ceci est une chaîne'  
foo (&$str);  
echo $str; // affiche 'Ceci est une chaîne, et un peu plus.'  
?>
```

# Fonctions PHP

## paramètres par défaut des fonctions

```
<?php  
function servir_apero ($type = "ricard")  
{  
    return "Servir un verre de $type.\n";  
}  
  
echo servir_apero();  
echo servir_apero("whisky");  
?>
```



**Valeurs par défaut : constantes**

**Paramètres avec valeurs par défaut sont passés après les autres**

# Fonctions PHP

## Nombre de paramètres variables

- Passer un tableau de variables (le plus simple)
- Utiliser les fonctions suivantes (en TP).
  - `func_num_args` ,
  - `func_get_arg`
  - `func_get_args` .

# Fonctions PHP

Exercices du soir :

- *Faire les fonctions implémentant les piles, les files, et une page les utilisant.*

# Inclusion de fichiers PHP

Pour inclure des bibliothèques de fonctions :

- Mots clefs
  - « require » « require\_once »  
« include » « include\_once »
- Ne se différencient que par leur gestion d'erreur :
- Inclut et exécute un fichier php.
  - « require » génère une erreur (le script s'interrompt)
  - « include » génère une alerte (le script continue).

# Inclusion de fichiers PHP

## Exemple

```
<?php  
// Avec une chaine constante :  
require 'prepend.php';  
  
// Avec une variable :  
require $somefile;  
  
// En pseudo fonction :  
require ('unfichier.txt');  
?>
```



AVANCEMENT

12 HEURES

# Variables prédéfinies PHP

Liste des variables prédéfinies (auto globales / super globales)

```
$GLOBALS // Ref sur chaque var globales. Clef = nom de variable.  
$_SERVER // variables fournies par httpd ou relatives au script courant.  
$_GET // variables fournies par http en méthode get.  
$_POST // variables fournies par http en méthode post.  
$_COOKIE // variables fournies par http dans les cookies.  
$_FILES // variables fournies par http suite à un chargement de fichier.  
$_ENV // variables fournies par l'environnement.  
$_REQUEST // variables fournies par tout mécanisme d'entrée.  
$_SESSION // variables enregistrée dans la session.
```

Connaître certaines variables prédéfinies: `phpinfo()`;

# Portée des variables en PHP

- Une variable utilisée dans une fonction est locale !
- La portée d'une variable concerne la totalité du script.
- Pour récupérer une variable globale dans une fonction :
  - utiliser le mot clef « global ».
  - utiliser le tableau \$globals
- Variables statiques : comme en C

# Formulaires HTML

Pourquoi faire ?

On utilise des formulaires dans le cas suivant :

- L'utilisateur doit fournir des infos au serveur.
- Le serveur doit traiter ces infos.
- Le serveur renvoie une réponse au client.

Un formulaire est mis en forme en HTML.

# Formulaires HTML

## Début d'explication

Les informations que fournit le formulaire doivent être identifiées pour le traitement. Elles portent un nom.

- Le contenu de l'information dépend du type d'information.
- Le serveur traite et renvoie une réponse au client.

Un formulaire est mis en forme en HTML.

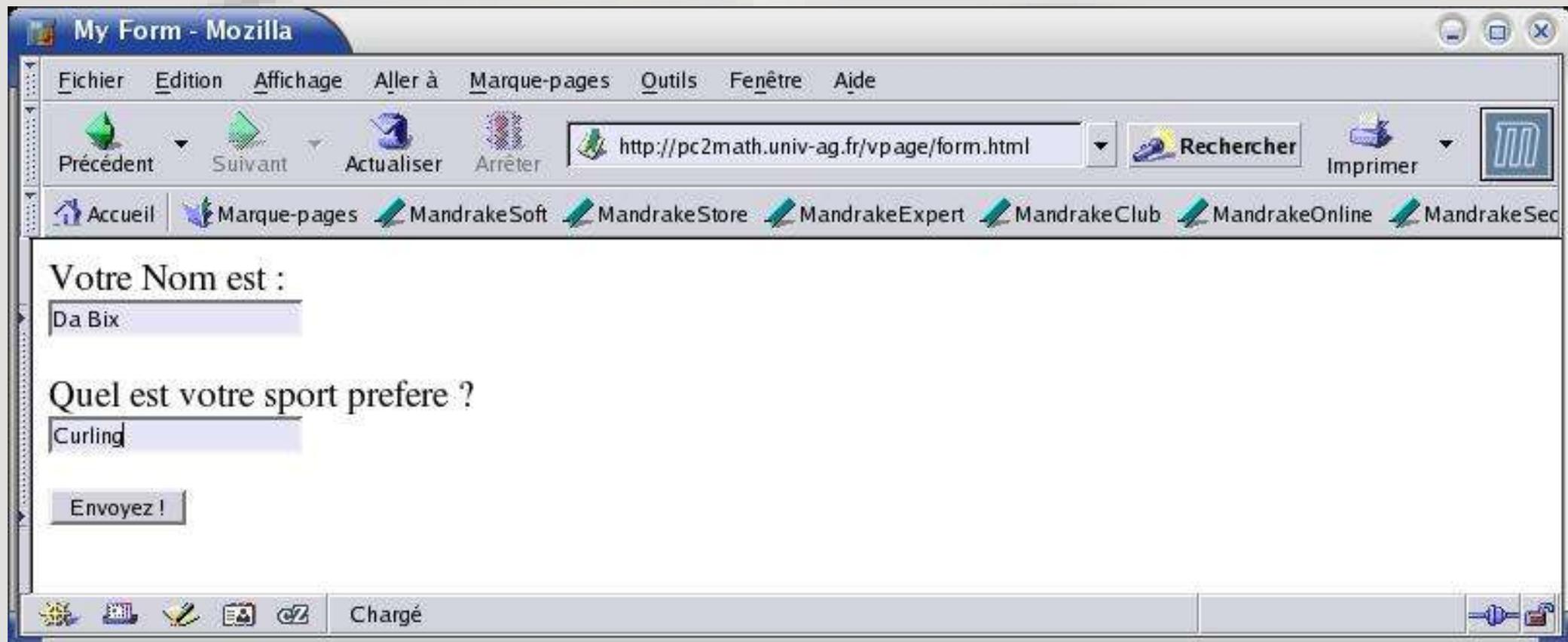
# Formulaires HTML

Ecriture d'un formulaire HTML :

```
<form action= "mon_script.php" method= "post">  
Votre Nom est : <br>  
<input type="text" name="YourName"> <p>  
  
Quel est votre sport prefere ? <br>  
<input type="text" name="FavoriteSport"> <p>  
  
<input type="submit" name="submit" value="Envoyez !">  
</form>
```

# Formulaires HTML

Résultat du formulaire précédent :



# Formulaires HTML

## Les <inputs> qui existent

Ils se distinguent par l'attribut type

- text : une ligne de texte
- password : idem mais caché.
- hidden : invisible (pour passer des variables).
- checkbox : selection multiple.
- radio : selection d'une possibilité parmi X.
- submit : bouton d'envoi de formulaire
- reset : bouton de remise dans l'état par défaut
- file : saisie d'un nom de fichier (et bouton parcourir...)

# Formulaires HTML

## Autres balises de formulaires

Attention, ce sont des balises

- `<select></select>` : liste de sélection déroulante.
  - `<option></option>` : options de liste déroulante.
- `<textarea></textarea>` : zone de texte multilignes...
- `<button></button>` : un bouton.

# Formulaires HTML

## Traitement du formulaire : Script PHP

- Le formulaire est envoyé à un script défini par l'attribut action de la balise <form> (ici mon\_script.php)
- Deux méthodes d'envoi : GET et POST défini par l'attribut method de la balise <form> (ici post)
- Le contenu du formulaire est accessible via une variable meta-globale (ici \$\_POST)
- Chaque champ du formulaire crée une variable portant son nom (attention aux checkbox...)
- Le résultat du traitement est renvoyé au client.

Utiliser `print_r($_POST)` pour debugger...

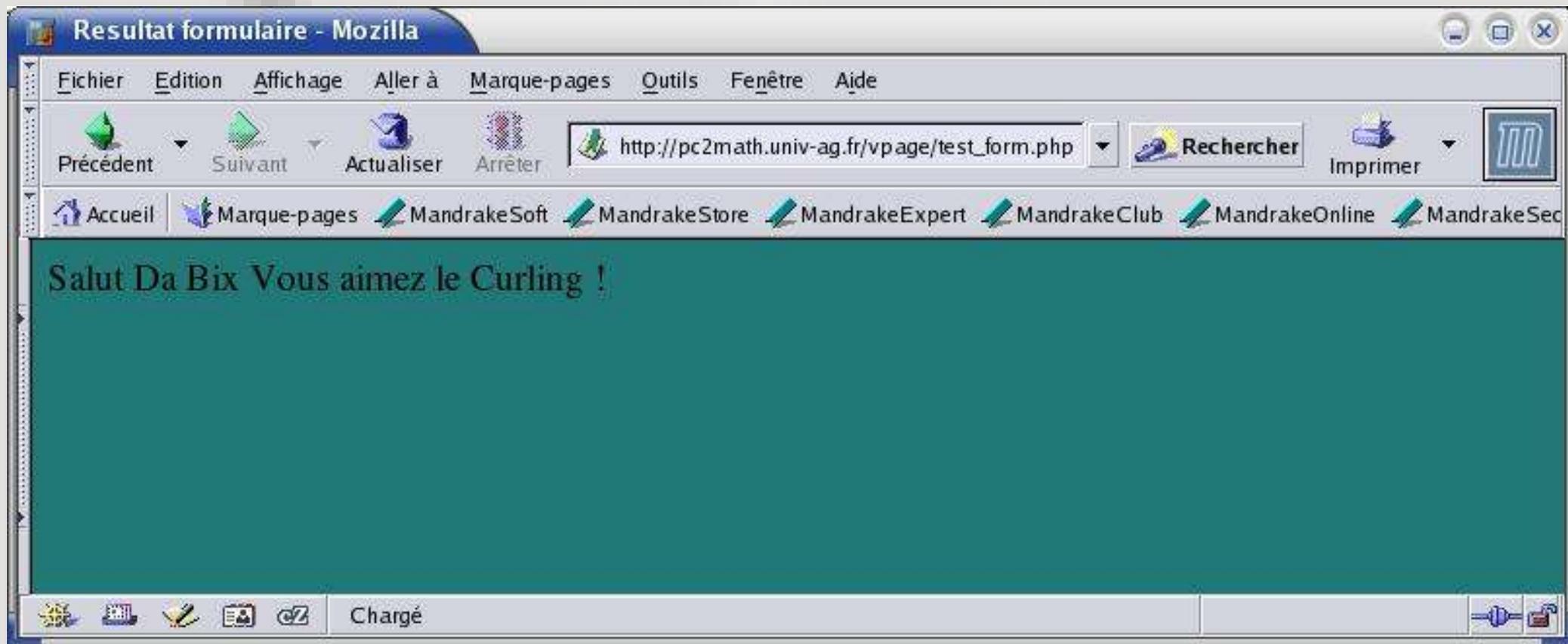
# Formulaires HTML

## Traitement du formulaire : Script PHP

```
<?php  
$name = $_POST["YourName"];  
$sport = $_POST["FavoriteSport"];  
printf ("Salut %s\n", $name);  
printf ("Vous aimez le %s !\n", $sport);  
?>
```

# Formulaires HTML

Résultat du traitement du formulaire précédent :



# Formulaires HTML

## Méthode Post et Méthode Get

- Méthode GET
  - Méthode par défaut.
  - Le formulaire est passé par ajout à l'url.
  - La requête a la forme : "GET /doc.html?var1=val1&var2=val2.... HTTP/1.0"
  - Le script php peut les récupérer avec la méta-globale `$_GET`
  - Bookmarkable.
  - Petits volumes de données.

# Formulaires HTML

## Méthode Post et Méthode Get

- Méthode POST
  - Le formulaire est passé après l'url.
  - Les données sont transmises sur l'entrée standard du script.
  - Un script php peut les récupérer avec `$_POST`
  - Non bookmarkables
  - Petits ou gros volumes de données.



AVANCEMENT

14 HEURES

# Sessions et Cookies

But : Conserver des informations lors de la navigation d'un client sur le serveur.

Exemples d'utilisation :

- Identification (accès contrôlé)
- Panier d'achat
- Affichage adapté au contexte.

En effet : http fonctionne en Mode non connecté.

# Cookies

- Qu'est ce ?
  - Un fichier créé par le navigateur sur le poste client, à la demande du serveur.
  - Sur la mdk9.2, les cookies de mozilla sont tous placés dans un fichier cookies.txt.
- Fonctionnement :
  - Le navigateur qui se connecte à un domaine envoie au serveur les cookies rattachés au domaine.
  - Le cookie a une durée de vie limitée. Il est supprimé lorsqu'il est périmé.

# Cookies

- Les Cookies sont créés coté client :
  - Problème d'acceptation.
- Php gère les cookies de façon simple.
  - Les cookies sont envoyés comme en-tête du document (avant tout autre chose).
- Envoyer un cookie :

*bool setcookie (string name, string value, int expire, string path, string domain, int secure)*

- Regarder les cookies : `$_COOKIE`



**Attention : cookies envoyés avant toute chose (doctype).**

# Cookies

Exemple :

```
<?php  
$duree = 24*60*60; // 1 journee en secondes !  
$value=$_SERVER["REMOTE_ADDR"];  
setcookie("moncookie",$value,time()+$duree,"/","127.0.0.1");  
?>  
  
<html> </body>  
  
vous venez de recevoir un cookie <br>  
<a href="lire_cookie.php"> Lire le cookie </a>  
  
</body></html>
```

# Cookies

Si vous avez besoin des cookies :

- vérifiez que vous pouvez les utiliser : Envoyez en un et regardez si il est reçu.

# Sessions

- Qu'est ce que c'est :
  - Un identifiant SID associé à un navigateur accédant à une page.
  - Le SID est défini par le serveur.
  - Coté serveur, on stocke pour chaque SID un ensemble de variables.
- Mode de passage du SID :
  - Par url. Peu sécurisé.
  - Par cookie. Risque de non acceptation des cookies.
  - Choix du mode de passage : php.ini
- Accès aux variables de sessions : `$_SESSION`

# Sessions

- Utiliser les sessions :

```
<?php  
session_start();  
if (!isset($_SESSION['compteur']))  
{ $_SESSION['compteur'] = 0; }  
else  
{ $_SESSION['compteur']++; }  
?>
```

# Cookies et sessions

- Authentification sécurisée (ma banque) :
  - Authentification par session.
    - limite les transmissions.
  - Passage du SID par cookie.
    - pas par l'url.
  - Le cookie à une durée de vie de 0
    - détruit quand on ferme le navigateur.
  - https : tous les échanges de données sont cryptés.
    - l'url passe quand même en clair

# AVANCEMENT

16 HEURES

# Chargement de fichiers en PHP

- Pour uploader un fichier : Partie HTML

- Le formulaire doit être encodé comme il faut :

```
<form enctype="multipart/form-data" action="upload_test.php" method=post>
```

- Utiliser le type file dans le formulaire.

```
<input type="file" name="sonfichier">
```



**Attention : légalité.**

# Chargement de fichiers en PHP

- Pour uploader un fichier : Partie Apache/PhP
  - Apache place le fichier sous un nom temporaire, dans le répertoire /tmp (ou définit dans php.ini).
  - La variable `$_FILES` est un tableau. La clef « name » du formulaire est associée à un tableau dont les clefs sont « name » et « tmp\_name ».
  - Le fichier doit être déplacé ou renommé dans le script. Sinon, il est effacé.

```
<?php
php print_r( $_FILES["sonfichier"]);
print_r( $_FILES["sonfichier"]["name"]);
print_r( $_FILES["sonfichier"]["tmp_name"]);
?>
```

# Chargement de fichiers en PHP

- Fonctions utiles :
  - `is_uploaded_file("nom_fichier");`
  - `move_uploaded_file($fichier_tampon, $upload_dir.$fichier_cible);`

# Manipulation de fichiers en PHP

Script exécuté par le serveur : les droits sur les fichiers sont ceux d'apache ou nobody.

- Obtenir des informations sur les fichiers locaux
  - `file_exists`, `is_dir`, `is_executable`, `is_file`, `is_link`, `is_readable`, `is_writable`
  - `stat`, `lstat`, `readlink`
  - `fileowner`, `filesize`, `filegroup`, `fileinode`, `fileperms`
  - `fileatime`, `filemtime`
  - `disk_free_space`, `disk_total_space`
  - `basename`, `dirname`,

# Manipulation de fichiers en PHP

Script exécuté par le serveur : les droits sur les fichiers sont ceux d'apache ou nobody.

- Quelques fonctions bash-like :
  - Créer : touch, mkdir, link, symlink, copy, tempnam
  - Déplacer, effacer : rename, rmdir
  - Modifier les droits : umask, chmod, chmod, chgroup

# Manipulation de fichiers en PHP

## Quelques fonctions C-like

- les « pointeurs de fichiers » utilisés sont de type *ressource*.
  - Ouverture fermeture : fopen, fclose, feof, fflush
  - Répertoires : opendir, readdir;
  - Déplacement : rewind, fseek
  - Ascii : fscanf, fprintf, fgets, fgetss (sécu tags)
  - Binaire : fread, fwrite.

# Manipulation de fichiers en PHP

## Quelques fonctions typiques php

- glob : Trouve les noms de fichiers correspondant à une règle.

```
foreach (glob("*.txt") as $filename) {echo $filename;}
```

- file : Met le contenu d'un fichier dans un tableau

```
foreach (file ("toto.txt", "./") as $ligne) {echo $ligne;}
```

- file\_get\_contents : idem file mais dans une chaîne.

```
echo file ("toto.txt", "./");
```

# Fichiers PHP

Afficher le contenu d'un répertoire

```
<?php  
$handle=opendir('.');  
echo "Pointeur de dossier: $handle <br>";  
echo "Fichiers: <br>";  
while ($file = readdir($handle))  
{  
    echo "$file <br>";  
}  
closedir($handle);  
?>
```



AVANCEMENT

18 HEURES

# Exemples PHP

Comment faire :

- Une page affichant toutes les images d'un répertoire.
- Une page affichant
  - un tas d'aperçus d'images.
  - Liées aux images.
  - commentaire pour chaque aperçu.
- Comment faire pour que seules certaines personnes ne puissent voir ces images. Images non confidentielles, juste privées.

# Exemples PHP

Quelques exemples

- Analyse du diaporama.
- Analyse de clock.php



# AVANCEMENT

20 HEURES

# PhP – Postgress

## Fonctions utiles

- `pg_connect` : se connecter sur un serveur postgress
- `pg_connection_status` : ....
- `pg_query` : faire une requete.
- `pg_fetch_array` : Lit une ligne de resultat dans un tableau.
- `pg_fetch_row()` : Lit une ligne dans un tableau (indices colonnes)
- `pg_close` : se déconnecter

# Exemple PhP – Postgress

## Exemples de connections

```
<?php  
$dbconn = pg_connect("dbname=marie");  
$dbconn2 = pg_connect("host=localhost port=5432 dbname=marie");  
$dbconn3 = pg_connect ("host=sheep port=5432 dbname=marie  
user=mouton password=baaaa");  
$conn_string = "host=sheep port=5432 dbname=test user=lamb  
password=bar";  
?>
```

# Exemple PhP – Postgress

## Exemples de requete

```
$db = pg_connect("...");  
$query = "SELECT * FROM friends";  
$result = pg_query($db, $query);  
  
$row=0;  
$numrows = pg_numrows($result);  
do {  
    $myrow = pg_fetch_row ($result,$row);  
    echo $myrow[0].$myrow[1]. $myrow[2];  
    $row++;  
} while ($row < $numrows);
```

# Sockets PHP

## Fonctions utiles

- `gethostbyname` : Renvoie le numéro IP d'une machine.
- `gethostbyaddr` : Renvoie le nom d'une machine.

# Sockets PHP

## Fonctions utiles client

- `socket_create` : Créer une socket
- `socket_connect` : Se connecter à une socket
- `pfsocketopen` : ouvrir une socket persistante.
- `socket_read` : Lecture dans une socket
- `socket_write` : Ecriture dans une socket
- `socket_close` : Fermer la socket.

# Sockets PHP

## Fonctions de remplacement client

- `fsockopen` : Créer une socket
- `fgets` : Lecture dans une socket
- `fputs` : Ecriture dans une socket

# Sockets PHP

## Fonctions utiles serveur

- `socket_create` : Créer une socket
- `socket_bind` : Donner un nom à la socket
- `socket_listen` : Attendre les connections
- `socket_accept` : Ecriture dans une socket
- `socket_close` : Fermer la socket.

# Exemple de socket

Récupération d'une page Web

*\$db*

# Génération d'images en lignes

Faut remplir ici... bibliothèque GD.

# Autres fonctions PHP

- strip\_slashes
- add\_slashes
- ???



AVANCEMENT

22 HEURES

# Sécurité

## Considérations de sécurité : Rappels

- Droits des scripts.
- Conservation de renseignements (cookies/session)
- Moyen de se loguer sur un SGBDR.
- Sécurisation des échanges client / serveur.



AVANCEMENT

24 HEURES