PHP

Hanafi YAKOUBEN

http://ceria.dauphine.fr/YakoubenHomePage/HyHomePage.html

Plan

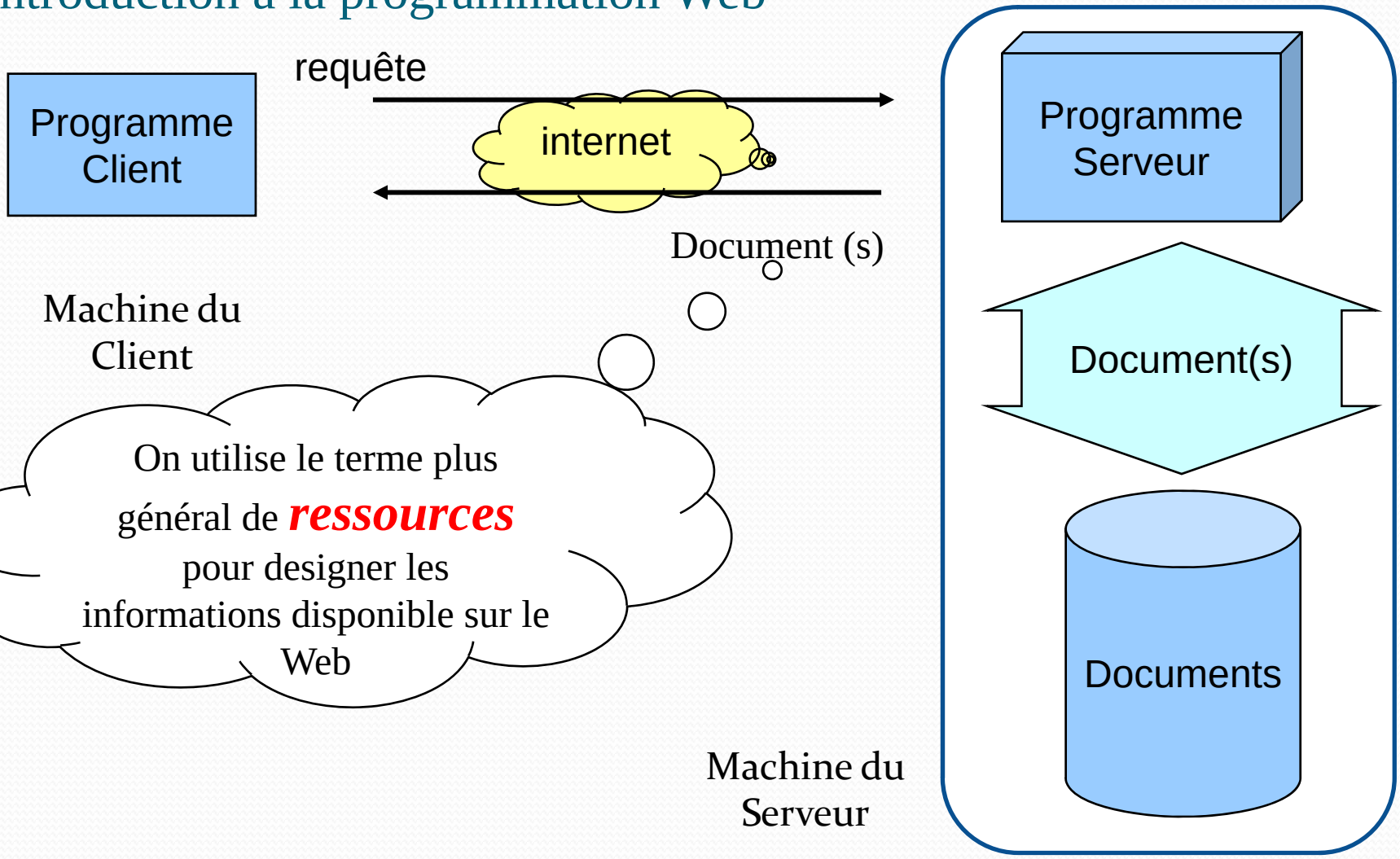
- **PARTIE 1 : Programmation Web**
- Introduction a la programmation Web (Serveur CGI et Client JavaScript)

- **PARTIE 2 : PHP**
- La programmation en PHP. Des centaines de fonctions détaillées et des exemples expliqués en détail.

- **PARTIE 3 : Introduction à SQL**
- Introduction au langage SQL.

- **PARTIE 4 : l'environnement MySQL/PHP**
- L'étude d'une application web réelle.

Introduction a la programmation Web



Exemple

- **Sur le serveur: code HTML**

```
<html>
<body>
<p>Bienvenue !</p>
</body>
</html>
```

- **Le client reçoit: code HTML identique**

```
<html>
<body>
<p>Bienvenue !</p>
</body>
</html>
```

Démo

Introduction a la programmation Web

1. **HTML n'est pas un langage de programmation**
   - Langage de structuration de l'information
   - Tendance actuelle: séparer l'information de son apparence
     - HTML contient l'information seulement
     - Feuille de style (CSS) contient la présentation de l'information

2. **Avantages de HTML**
   - Développement facilité
   - Intégration multimédia
     - Texte, images, liens, vidéo…

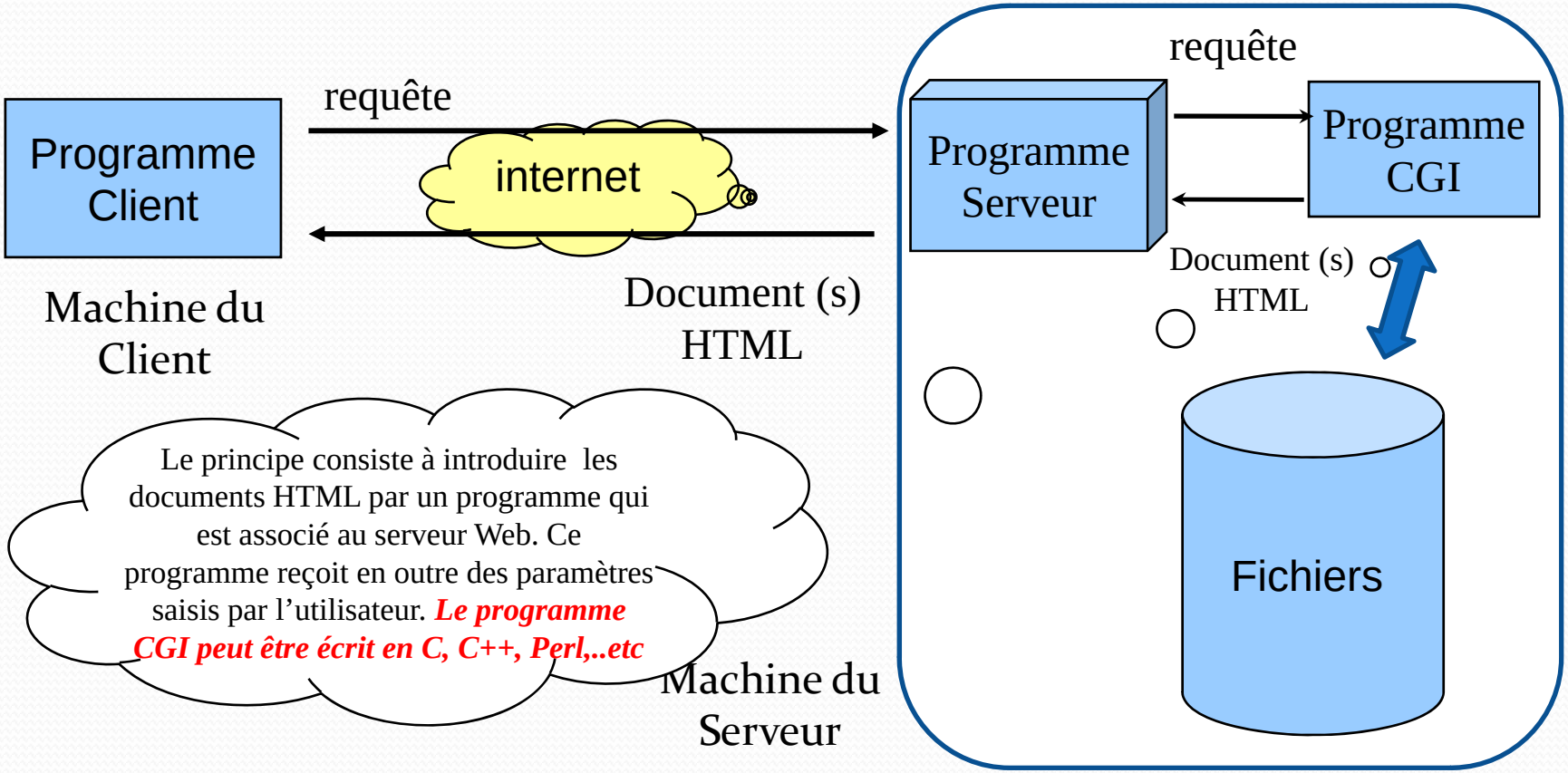
3. **Inconvénients de HTML**
   - Présentation statique de l'information
   - Pas d'interaction avec l'utilisateur

Script et page dynamique

- Les scripts côté client (comme JavaScript)
  - rendent les pages dynamiques
  - Modification dynamique de la présentation du contenu
  - Réponse aux actions de l'utilisateur
  - Vérification des formulaires
  - …
- Ils répondent au besoin d'interactivité
  - Mais pas au besoin de dynamisme du contenu
- D'où la nécessité de scripts côté serveur (comme PHP)
  - Permettent la modification dynamique du contenu
  - Interagissent avec des bases de données

Introduction a la programmation Web

- Architecture ***Common Gateway Interface*** (CGI)



- **Principe de base du CGI**

Exécution en trois phases:

1. ***Requête du Client au serveur*** : le programme serveur récupère les informations transmises par le navigateur , c'est-à-dire les nom du programme CGI accompagné, le plus souvent, de paramètres saisis par l'utilisateur dans un formulaire;
2. ***Exécution du programme CGI***: le serveur déclenche l'exécution du programme CGI, en lui fournissant les paramètres reçus;
3. ***Transmission du programme HTML***: le programme CGI renvoie l' exécution au serveur sous forme d'un document HTML, le serveur se contente alors de faire suivre au Client

Quelques limites de la programmation CGI

- **Passage de paramètres**: le passage des paramètres au programme CGI est relativement pénible à gérer. Idéalement, on souhaite disposer, directement dans le programme, des variables correspondant aux champs du formulaire HTML: ***c'est ce que fait PHP***

- **Faibles intégration avec HTML**: la programmation CGI n'est pas une extension de HTML qui permettrait de recourir à ce langage pour la partie des page qui correspond à du texte fixe.

- **Sessions**: le serveur ne mémorise pas les demandes effectuées par un client particulier, et ne peut donc pas tenir compte de l'historique des échanges pour améliorer la communication avec le client.

Le langage PHP

PHP

- Il a été créé en 1994 par Rasmus Lerdorf pour les besoins des pages web personnelles (livre d'or, compteurs, etc.). A l'époque, **PHP signifiait *Personnal Home Page*.**

- C'est un langage incrusté au HTML et **interprété** (PHP3) ou **compilé** (PHP4) côté serveur. Il dérive du C et du Perl dont il reprend la syntaxe. Il est extensible grâce à de nombreux modules et son code source est ouvert. Comme il supporte tous les standards du web et qu'il est gratuit, il s'est rapidement répandu sur la toile.

- En 1997, PHP devient un projet collectif et son interpréteur est réécrit par Zeev Suraski et Andi Gutmans pour donner la version 3 qui s'appelle désormais ***PHP : Hypertext Preprocessor*** (acronyme récursif à l'exemple du système Open Source *Linux : Is Not UniX*).

- Il existe par ailleurs des applications web prêtes à l'emploi (PHPNuke, PHP SPIP, PHPSlash…) permettant de monter facilement et gratuitement son portail. En juillet 2000 plus de 300.000 sites tournaient déjà sous PHP !

Pourquoi PHP?

- PHP
  - Exécuté côté serveur
  - Code non lisible par l'utilisateur
  - Ne permet pas de modifier la structure de la page
  - Interactivité plus difficile avec l'utilisateur
  - Rechargement (contact avec le serveur) nécessaire
  - Accès à une base de données
  - Enregistrement et restitution de données
  - Ne peut pas fonctionner «hors ligne»

PHP/Serveur

- PHP permet d'accéder à des bases de données
  - Différents types de bases sont possibles
  - Par exemple: MySQL, PostgreSQL
- PHP nécessite un serveur pour fonctionner
  - Par exemple: serveur **L**AMP, **W**AMP ou **M**AMP
  - **L**inux, **W**indows ou **M**ac OS: système d'exploitation
  - **A**pache: serveur HTTP
  - **M**ySQL: gestionnaire bases de données
  - **P**HP: interpréteur script
- Solution simple sous Windows: **EasyPHP**

PHP

- Page statique et page dynamique
- Page HTML = page statique
  - Extension «**.htm**» **ou** «**.html**»
  - Transmise directement au client
  - Code source lisible par le client
- Page PHP = page dynamique
  - Extension «**.php**», «**.php3**» **ou** «**.php4**»
  - *Conseil: utiliser plutôt l'extension «**.php**»*
  - Interprétée par le serveur
  - Génère du HTML transmis au client
  - Code source non lisible par le client

PHP, intégration d'un script dans une page

- Le code source php est directement insérer dans le fichier html grâce au conteneur de la norme XML :

Balise de début: **<?php**

Balise de fin: **?>**

Démo

PHP, intégration d'un script dans une page

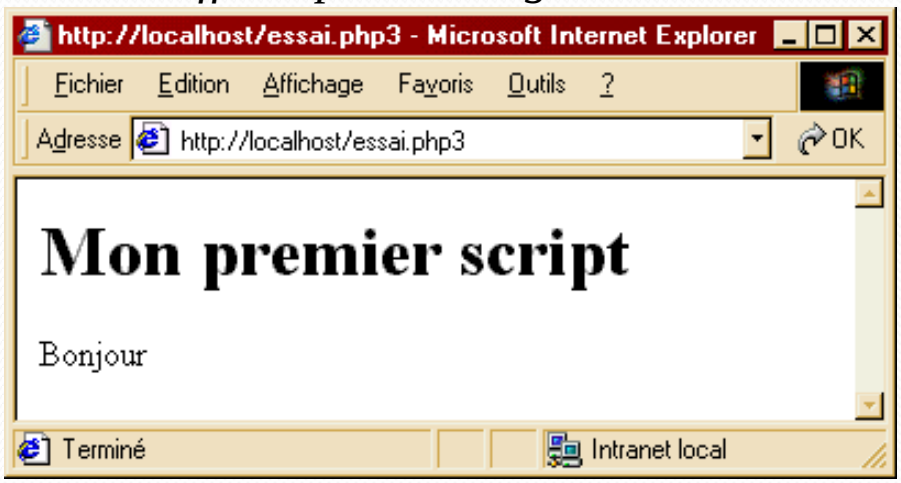
Exemple de script, code source (côté serveur) :

```
<html>
<body>
<h1>Mon premier script</h1>
<?php echo ''Bonjour\n''; ?>
</body>
</html>
```

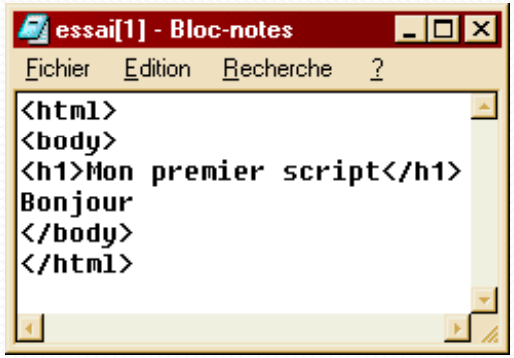
Autre écriture du même script :

```
<?php
echo ''<html>\n<body>\n'';
echo ''<h1>Mon premier script</h1>\n'';
echo ''Bonjour\n'';
echo ''</body>\n</html>\n'';
?>
```

Résultat affiché par le navigateur :



Code source (côté client) de la page essai.ph résultant du script



PHP, les commentaires

- Un script php se commente comme en C :
- *Exemple :*

```
<?php

// commentaire de fin de ligne

/* commentaire
sur plusieurs
lignes */

# commentaire de fin de ligne comme en Shell
?>
```

Tout ce qui se trouve dans un commentaire est ignoré. Il est conseillé de commenter largement ses scripts.

PHP, variables, types et operateurs

- Le typage des variables est implicite en php. Il n'est donc pas nécessaire de déclarer leur type au préalable ni même de les initialiser avant leur utilisation.

- Les identificateurs de variable sont précédés du symbole « **$** » (dollars). Exemple : **$toto**.

- Les variables peuvent être de type entier (**integer**), réel (**double**), chaîne de caractères (**string**), tableau (**array**), objet (**object**), booléen (**boolean**).

- Il est possible de convertir une variable en un type primitif grâce au cast(1) (comme en C).
- *Exemple :*

  $str = ”12”; // $str vaut la chaîne **‘’12”**

  $nbr = (int)$str; // $nbr vaut le nombre **12**
- (1) : Le cast est une conversion de type. L'action de caster consiste en convertir une variable d'un type à un autre.

PHP, variables, types et operateurs

- Quelques fonctions :
  - **empty($var)** : renvoie vrai si la variable est vide
  - **isset($var)** : renvoie vrai si la variable existe
  - **unset($var)** : détruit une variable
  - **gettype($var)** : retourne le type de la variable
  - **settype($var, ”type”)** : convertit la variable en type **type** (cast)
  - **is_long()**, **is_double()**, **is_string()**, **is_array()**, **is_object()**, **is_bool()**, **is_float()**, **is_numeric()**, **is_integer()**, **is_int()…**

- Une variable peut avoir pour identificateur la valeur d’une autre variable.
- *Syntaxe :* **${$var} = valeur;**
- *Exemple :*
  - **$toto = ‘’foobar”;**
  - **${$toto} = 2002;**
  - **echo $foobar;** // la variable **$foobar** vaut **2002**

PHP, variables, types et operateurs

- La portée d'une variable est limitée au bloc dans lequel elle a été crée. Une variable locale à une fonction n'est pas connue dans le reste du programme. Tout comme une variable du programme n'est pas connue dans une fonction. Une variable créée dans un bloc n'est pas connue dans les autres blocs, mêmes supérieurs.

- Opérateurs arithmétiques :
  + (addition), **-** (soustraction), ***** (multiplié), / (divisé), **%** (modulo), **++** (incrément), **--**(décrément). Ces deux derniers peuvent être pré ou post fixés

- Opérateurs d'assignement :
  = (affectation), ***=** (**$x*=$y** équivalent à **$x=$x*$y**), /=, +=, -=, **%=**

- Opérateurs logiques :
  and, **&&** (et), **or**, || (ou), **xor** (ou exclusif), **!** (non)

- Opérateurs de comparaison :
  == (égalité), < (inférieur strict), <= (inférieur large), >, >=, **!**= (différence)

PHP, variables, types et operateurs

- Signalons un opérateur très spécial qui équivaut à une structure conditionnelle complexe *if then else* à la différence qu'il renvoie un résultat de valeur pouvant ne pas être un booléen : l'opérateur ternaire.
- *Syntaxe :*

  (condition)?(expression1):(expression2);
- Si la **condition** est vrai alors évalue et renvoie l'**expression1** sinon évalue et renvoie l'**expression2**.

- *Exemple :*

  $nbr = ($toto>10)?($toto):($toto%10);

  Dans cet exemple, la variable **$nbr** prend **$toto** pour valeur si **$toto** est strictement supérieur à **10**, sinon vaut le reste de la division entière de **$toto** par **10**.

PHP, les constantes

- L'utilisateur peut définir des constantes dont la valeur est fixée une fois pour toute. Les constantes ne portent pas le symbole **$** (dollars) en début d'identificateur et ne sont pas modifiables.

- **define(''var",valeur)** : définit la constante **var** (sans **$**) de valeur **valeur**

- *Exemple 1 :*

```
define(''author","Foobar");
echo author;                    // affiche 'Foobar'
```

- *Exemple 2 :*

```
define(MY_YEAR,1980);
echo MY_YEAR;                   // affiche 1980
```

- Contrairement aux variables, les identificateurs de constantes (et aussi ceux de fonction) ne sont pas sensibles à la casse.

PHP, les références

- On peut à la manière des pointeurs en C faire référence à une variable grâce à l'opérateur **&** (ET commercial).

- *Exemple 1 :*

```
$toto = 100;      // la variable $toto est initialisée à la valeur 100
$foobar = &$toto; // la variable $foobar fait référence à $toto
$toto++;                    // on change la valeur de $toto
echo $foobar;               // qui est répercutée sur $foobar qui vaut alors 101
```

- *Exemple 2 :*

```
function change($var) {
    $var++;       // la fonction incrémente en local l'argument
}
$nbr = 1;         // la variable $nbr est initialisée à 1
change(&$nbr);            // passage de la variable par référence
echo $nbr;        // sa valeur a donc été modifiée
```

PHP, les booléens

- Les variables booléennes prennent pour valeurs **TRUE** (vrai) et **FALSE** (faux). Une valeur est interprétée comme étant vraie si elle est différente de 0 ou de la chaine vide, et fausse sinon

- Exemple :

```
if(0) echo 1;                    // faux
if("") echo 2;        // faux
if("0") echo 3;       // faux
if("00") echo 4;
if('0') echo 5;       // faux
if('00') echo 6;
if(" ") echo 7;
```

Cet exemple affiche **467**. Donc l'espace ou la chaîne **''00''** ne sont pas considérés castés en **FALSE**.

PHP, les chaines de caractères

- Une variable chaîne de caractères n'est pas limitée en nombre de caractères. Elle est toujours délimitée par des simples quotes ou des doubles quotes.

- *Exemples :*

```
$nom = "Yakouben";
$prenom = 'Hanafi';
```

- Les doubles quotes permettent l'évaluation des variables et caractères spéciaux contenus dans la chaîne (comme en C ou en Shell) alors que les simples ne le permettent pas.

- *Exemples :*

```
echo "Nom: $nom";      // affiche Nom: Yakouben
echo 'Nom: $nom';      // affiche Nom: $nom
```

- Quelques caractères spéciaux : **\n** (nouvelle ligne), **\r** (retour à la ligne), **\t** (tabulation horizontale), \\ (antislash), **\$** (caractère dollars), **\"** (double quote).

- *Exemple :*

```
echo "Hello Word !\n";
```

PHP, les chaines de caractères

- Opérateur de concaténation de chaînes : **.** (point)

- *Exemple 1 :*

```
$foo = ''Hello'';
$bar = ''Word'';
echo $foo.$bar;                    // affichage de Hello World
```

- *Exemple 2 :*

```
$name = ''Henry'';
$whoiam = $name.''IV'';
```

- *Exemple 3 :*

```
$out = 'Patati';
$out .= '' et patata...'';
```

PHP, affichage

- Les fonctions d'affichage :

echo() : écriture dans le navigateur

print() : écriture dans le navigateur

printf([$format, $arg1, $arg2]) : écriture formatée comme en C, 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);
```

PHP, tableaux indicés

- Une variable tableau est de type **array**. Un tableau accepte des éléments de tout type. Les éléments d'un tableau peuvent être de types différents et sont séparés d'une virgule.

- Un tableau peut être initialisé avec la syntaxe **array**.
- *Exemple :*

```
$tab_colors = array('red', 'yellow', 'blue', 'white');
$tab = array('foobar', 2002, 20.5, $name);  // type mixte
```

- Mais il peut aussi être initialisé au fur et à mesure.
- *Exemples :*

```
$villes[0] = ''Paris";                $villes[] = ''Paris";
$villes[1] = ''Londres";              $villes[] = ''Londres";
$villes[2] = ''Lisbonne";             $villes[] = ''Lisbonne";
//affectation automatique d'un indice à un nouvel élément du tableau
```

- L'appel d'un élément du tableau se fait à partir de son indice (dont l'origine est zéro comme en C**)**.
- *Exemple :* **echo $tab[10];** // pour accéder au 11ème élément

PHP, tableaux indicés

- Parcours d'un tableau.

  $tab = array('Hugo', 'Jean', 'Mario');

- *Exemple 1 :*

```
$i=0;
while($i <= count($tab)) {          // count() retourne le nombre d'éléments
  echo $tab[$i].'\n';
  $i++;
}
```

- *Exemple 2 :*

```
foreach($tab as $elem) {
  echo $elem."\n";
}
```

La variable **$elem** prend pour valeurs successives tous les éléments du tableau **$tab**.

PHP, tableaux associatifs

- Un tableau associatif est appelé aussi *dictionnaire* ou *hashtable*. On associe à chacun de ses éléments une clé dont la valeur est de type chaîne de caractères. Elle doit être unique pour l'ensemble du tableau.

- L'initialisation d'un tableau associatif est similaire à celle d'un tableau normal.

- *Exemple 1 :*

  $personne[''Nom''] = ''César'';

  $personne[''Prénom''] = ''Jules'';

  On peut utiliser *array* pour initialiser ce tableau

- *Exemple 2 :*

```
$personne = array(
''Nom'' => ''César'',
''Prénom'' => ''Jules''
);
```

- Ici à la clé **''Nom''** est associée la valeur **''César''**.

PHP, tableaux associatifs

- Parcours d'un tableau associatif.

```
$personne = array(
"Nom" => "César",
"Prénom" => "Jules");
```

- *Exemple 1 :*

```
Do
{ echo "Clé=Key($personne ) .valeur=current($personne ) "}
While (next($personne ));
```

- *Exemple 2 :*

```
foreach($personne as $key => $elem) {
  echo "$key : $elem";
}
```

- Ici on accède simultanément aux clés et aux éléments.

PHP, tableaux associatifs

Quelques fonctions alternatives pour le parcours de tableaux (normaux ou associatifs) :

reset($tab) : place le pointeur sur le premier élément

current($tab) : retourne la valeur de l'élément courant

next($tab) : place le pointeur sur l'élément suivant

prev($tab) : place le pointeur sur l'élément précédant

each($tab) : retourne la paire clé/valeur courante et avance le pointeur

Exemple :

```
$colors = array(''red", "green", "blue");
$nbr = count($colors);
reset($colors);
for($i=1; $i<=$nbr; $i++) {
        echo current($colors)."<br />";
        next($colors);
}
```

PHP, les fonctions

- Comme tout langage de programmation, php permet l'écriture de fonctions.

- Les fonctions peuvent prendre des arguments dont il n'est pas besoin de spécifier le type. Elles peuvent de façon optionnelle retourner une valeur.

- L'appel à une fonction peut ne pas respecter son prototypage (nombre de paramètres). Les identificateurs de fonctions sont insensibles à la casse.

- *Exemple :*

```
function mafonction($toto) {
    $toto += 15;
    echo ''Salut !'';
    return ($toto+10);
}
$nbr = MaFonction(15.1);
/* retourne 15.1+15+10=40.1, les majuscules n'ont pas d'importance */
```

PHP, les fonctions

- On peut donner une valeur par défaut aux arguments lors de la déclaration de la fonction. Ainsi, si un argument est « oublié » lors de l'appel de la fonction, cette dernière lui donnera automatiquement une valeur par défaut décidée à l'avance par le programmeur.
- *Exemple :*

```
function Set_Color($color=‘’black”) {
    global $car;
    $car[‘’color”] = $color;
}
```

- Forcer le passage de paramètre par référence (équivalent à user de **global**):
- *Exemple :*

```
function change(&$var) { // force le passage systématique par référence
    $var += 100;          // incrémentation de +100
}
$toto = 12;       // $toto vaut 12
change($toto); // passage par valeur mais la fonction la prend en  référence
echo $toto;       // $toto vaut 112
```

PHP, les fonctions

Grâce à une petite astuce, il est possible de faire retourner plusieurs valeurs d'une fonction. En retournant un tableau ayant pour éléments les variables à retourner. Dans l'appel de la fonction, il faudra alors utiliser la procédure **list()** qui prend en paramètre les variables à qui ont doit affecter les valeurs retournées. On affecte à **list()** le retour de la fonction.

Exemple :

```
function trigo($nbr) {
    return array(sin($nbr), cos($nbr), tan($nbr));          // retour d'un tableau
}
$r = 12;
list($a, $b, $c) = trigo($r);    /* affectation aux variables $a,$b et $c des éléments du tableau retourné par la fonction trigo */
echo ''sin($r)=$a, cos($r)=$b, tan($r)=$c";        // affichage des variables
```

Cet exemple affichera ceci :

sin(12)=-0,5365729180, cos(12)=0,8438539587, tan(12)=-0,6358599286

PHP, les fonctions

Il est possible de créer dynamiquement des fonctions. Pour les déclarer, on affecte à une variable chaîne de caractères le nom de la fonction à dupliquer. Puis on passe en argument à cette variable les paramètres normaux de la fonction de départ.

Exemple :

```
function divers($toto) {
        echo $toto;
}
$mafonction = "divers";
$mafonction("bonjour !");        // affiche 'bonjour !' par appel de divers()
```

PHP, structures de contrôle

- Structures conditionnelles (même syntaxe qu'en langage C) :

```
if( … ) {
    …
} elseif {
    …
} else {
    …
}

switch( … ) {
    case … : { … } break
    …
    default : { … }
}
```

PHP, structures de contrôle

- L'instruction **break** permet de quitter prématurément une boucle.
- *Exemple :*

```
while($nbr = $tab[$i++]) {
    echo $nbr."<br />";
    if($nbr == $stop)
      break;
}
```

- L'instruction **continue** permet d'éluder les instructions suivantes de l'itération courante de la boucle pour passer à la suivante.
- *Exemple :*

```
for($i=1; $i<=10; $i++) {
    if($tab[$i] == $val)
      continue;
    echo $tab[$i];
}
```

PHP, inclusions

On peut inclure dans un script php le contenu d'un autre fichier.

require insert dans le code le contenu du fichier spécifié même si ce n'est pas du code php. Est équivalent au préprocesseur *#include* du C.

Exemple :

require(''**fichier.php**'');

include évalue et insert à chaque appel (même dans une boucle) le contenu du fichier passé en argument.

Exemple :

include(''**fichier.php**'');

- Comment ouvrir un fichier ?

La fonction **fopen** est utilisé pour ouvrir un fichier

Exemple:

```
<?php
   $f=fopen("exemple.txt","r");
   ?>
```

Si la fonction fopen() échoue lors de l'ouverture du fichier, celle ci retourne 0 (False)

PHP, Fichiers

- Les différents modes:
- **'r'** Ouvre en lecture seule : Le pointeur est placé au début du fichier.
  'r+' Ouvre en lecture/écriture : Le pointeur est placé au début du fichier.
  'w' Ouvre en écriture seule : Le pointeur est placé au début du fichier. Réduit la taille du fichier à 0. Tentative de création si celui ce n'existe pas.
  'w+' Ouvre en lecture/écriture : Le pointeur est placé au début du fichier. Réduit la taille du fichier à 0. Tentative de création si celui ce n'existe pas.

PHP, Fichiers

- **'a'** Ouvre en écriture seule : Le pointeur est placé au début du fichier. Tentative de création si celui ce n'existe pas.
  'a+' Ouvre en lecture/écriture : Le pointeur est placé à la fin du fichier. Tentative de création si celui ce n'existe pas.
  'x' Crée et ouvre le fichier en lecture seule : Le pointeur est placé au début du fichier. Si le fichier existe déjà, fopen va échouer.
  'x+' Crée et ouvre le fichier en lecture/écriture : Le pointeur est placé au début du fichier. Si le fichier existe déjà, fopen va échouer.

- Exemple:

```
<html>
<body>
<?php
if (!($f=fopen("exemple.txt","r")))
exit("Unable to open file!");
?>
</body>
</html>
```

- L'exemple ci-desus génère un message si la fonction fopen() est incapable d'ouvrir le fichier spécifié.

- Fermer un fichier:

Synthax:

```
<?php
  fclose($f);
  ?>
```

- Écrire dans un fichier:

L'écriture dans un fichier texte se fait avec la fonction fwrite(). Pour faire des retour à la ligne vous devez utiliser : "\r\n"

PHP, Fichiers

```php
<?php
$f = 'exemple.txt';
$text = "ma chaine de caractères";
$handle = fopen("$f","r");
// regarde si le fichier est accessible en écriture
if (is_writable($f)) {
// Ecriture
   if (fwrite($handle, $text) === FALSE) {
     echo 'Impossible d\'écrire dans le fichier '.$f.";
     exit;
   }
   echo 'Ecriture terminé';

   fclose($handle);
}
else {
     echo 'Impossible d\'écrire dans le fichier '.$f.";
   }
?>
```

- Compter le nombre de lignes:

  A l'aide de 2 fonctions. L'une place le fichier dans un tableau et l'autre pour compter les lignes du tableau.
- Exemple:

```
<?php

    $f = 'exemple.txt';
    if(file_exists($f)) {
        $tab = file($f);            // place le fichier dans un tableau
        $nb = count($tab);     // compte le nombre de ligne
        echo $nb;                   // Affiche le résultat
    }

    ?>
```

- Trouver la fin d'un fichier:
- La fonction feof() est utilisé pour déterminer si le pointeur est placé à la fin du fichier ou lister un fichier jusqu'au dernier caractère
- Exemple:

```
<?php
  if (feof($f))
  echo 'Fin du fichier';
  ?>
```

PHP, Fichiers

- Récupérer le contenu:

Nous allons nous servir de la fonction file_get_contents pour récupérer l'intégralité du fichier

- Exemple:

```
<?php

   $contents = file_get_contents("url du fichier");


  // Ou Avec fopen, fread, fclose

   $fd = fopen($url, 'r');                    // Ouverture du fichier en lecture seule
   while(!eof($fd))
   {
     $contents .= fread($fd, 8192);        //Récupère le contenu du fichier
                                           // et le place dans une chaine
                                           // de caractères  jusqu'a length=8192
   }
   fclose($fd);                          //Fermeture du fichier

   ?>
```

PHP, Fichiers

- Comment lire caractère par caractère ?

La fonction fgetc() est utilisé pour lire les caractères dans un fichier

- Exemple:

```
<?php
  if (!($f=fopen("exemple.txt","r")))
  exit("Impossible d'ouvrir le fichier.");
  while (!feof($f))
  {
  $texte=fgetc($f);
  echo $texte;
  }
  fclose($f);
?>
```

PHP, Programmation Orientée-Objet

- Une classe est une collection de variables et de fonctions qui fonctionnent avec ces variables. Une classe est définie en utilisant la syntaxe suivante :
- *Exemple :*

```
<?php
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 $this faisant référence à l'objet est obligatoire
    function Set_Couleur($couleur) {
        $this->couleur = $couleur;
    }
}
$mavoiture = new Voiture();              // création d'une instance
$mavoiture->Set_Couleur(''blanche'');    // appel d'une méthode
$coul = $mavoiture->couleur;
?>
```

- Question: est ce qu'on peut découper la définition d'une class en plusieurs fichiers ou en plusieurs blocs?
- Exemple:

```
<?php
   class test {
   ?>
   <?php
      function test() {
         echo 'OK';
      }
   }
   ?>
```

```
<?php
class test {
   function test()
{
      ?>
      <?php
      print 'Ok';
   }
}
?>
```

NE POUVEZ PAS
per la définition
classe en plusieurs
ers. De la même
vous **NE POUVEZ**
uper la définition
ne classe en de
les blocs, à moins
coupure ne soit à
ntérieure de la
claration d'une
méthode.

PHP, Programmation Orientée-Objet

- Le système de classes de php est très succinct, il ne gère que l'héritage simple.

- *Exemple :*

```
class Voituredeluxe extends Voiture {   // déclaration de la sous classe
    var $couleur;
    function voituredeluxe() {          // constructeur
      $this->Voiture();
    }
    function Set_Couleur($couleur) {
      $this->couleur = $couleur;
    }
    function Get_Couleur() {
      return $this->couleur;
    }
}
```

Un **constructeur**, peut être utilisée pour initialiser les attributs d'un objet. Il doit avoir soit le même nom que la classe, soit le nom réservé ***__construct***

La nouvelle classe **Voituredeluxe** hérite de tous les attributs et méthodes de la classe parente **Voiture** dont elle est une extension (**extends**). Il est possible de surcharger les méthodes, d'en déclarer de nouvelles, etc.

- Quelques fonctions :
- **get_declared_classes**() : retourne un tableau listant toutes les classes définies
- **class_exists($str)** : vérifie qu'une classe dont le nom est passé en argument a été définie
- **get_class($obj)**, **get_parent_class** : retournent le nom de la classe de l'objet **$obj**
- **get_class_methods($str)** : retourne les noms des méthodes de la classe **$str** dans un tableau

- **get_class_vars($str)** : retourne les valeurs par défaut des attributs de la classe **$str** dans un tableau associatif
- **get_object_vars($obj)** : retourne un tableau associatif des attributs de l'objet **$obj** les clés sont les noms des attributs et les valeurs, celles des attributs si elles existent
- **is_subclass_of($obj,$str)** : détermine si l'objet **$obj** est une instanciation d'une sous-classe de **$str**, retourne VRAI ou FAUX
- **method_exists($obj,$str)** : vérifie que la méthode **$str** existe pour une classe dont **$obj** est une instance, retourne VRAI ou FAUX

Comment gérer une session Web?

- Insérer à l'identifiant de session dans toutes ;
  - Les URL des pages transmises au client
  - Dans tous les formulaires.

- Inconvénients:
  - Lourde à mettre en œuvre
  - Fragile, il suffit que l'internaute accède à ne serait-ce qu'une seul page d'un autre site pour que l'identifiant de session soit perdu.

Créer un ou plusieurs Cookies

- Pour stocker l'identifiant de session, éventuellement d'autre informations du coté programme client.

- Noté que les cookies ne font pas partie du protocole HTTP

- **session_start()** : initialise des informations de la session.
  - Si aucune session n'existe, un identifiant est engendré et transmis dans un cookie.
  - Sinon la fonction instancie toutes les variables qui lui sont liées.
  - Doit être appelée au début de tout script utilisant les sessions.
- **session_destroy()** : détruit les données de session et ferme la session
- **Session_id()**: renvoie l'identifiant de 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 et les bases de données

Introduction à SQL

PHP et les bases de données

- SQL: *Structured Query Language*
  - Langage universel des bases de données
  - Permet l'interrogation d'une base
  - Permet la modification d'une base
- MySQL
  - Gestionnaire de bases de données
  - Commandé par le langage SQL
- PHP peut interagir avec MySQL
  - Envoi de requêtes SQL
  - Récupération du résultat
  - Ajout de données
- phpMyAdmin
  - Interface Web pour accéder à MySQL
  - Facilite la gestion des bases
  - Permet de tester ses requêtes

Bases de données: vocabulaire (attribut, tuple, Clé…etc)

Titre	Année	Id_realisateur	Genre
Impitoyable	1992	20	western
Van Gogh	1990	29	drame
Kagemusha	1980	68	Drame
Les pleins pouvoirs	1997	20	policier

Table Film

Id	Nom	Prénom	Année_Naissance
20	Eastwood	Clint	1930
21	Hackman	Gene	1930
29	Pialat	Maurice	1925
30	Dutronc	Jacques	
68	Kurosawa	Akira	1910

Table Artiste

Titre	Id_acteur	Nom_rôle
Impitoyable	20	William Munny
Impitoyable	21	Little Bill Dagget
Van Gogh	30	Van Gogh
Les pleins pourvoirs	21	Le président

Table Rôle

Bases de données: vocabulaire (attribut, tuple, Clé primaire)

- On appelle **attributs** le nom des colonnes qui représentent les constituants de l'entité. Un attribut (une colonne) est repéré par un nom et un domaine de définition, c'est-à-dire l'ensemble des valeurs qu'il peut prendre.
- On appelle **tuple** (ou n-uplet) une ligne du tableau.
- On appelle **clé primaire** est un ensemble minimal d'attributs permettant de désigner de façon unique un **tuple** dans une table (une relation).

bases de données: vocabulaire SQL

- La structure de base d'une requête SQL comprend trois clauses SELECT, FROM et WHERE.
- **SELECT** indique la liste des attributs constituant le résultat
- **FROM** indique la (ou les) tables dans lesquelles on trouve les attributs utiles à la requête.
- **WHERE** indique les conditions que doivent satisfaire les lignes de la table pour faire partie du résultat. La clause WHERE est optionnelle: toutes les lignes de la table sont sélectionnées si elle est omise.

Requêtes SQL: clause SELECT avec WHERE

- Exemple :

SELECT * FROM Film

Titre	Année	Id_realisateur	Genre
Impitoyable	1992	20	western
Van Gogh	1990	29	drame

SELECT Titre , Année FROM Film WHERE Année=1992

Titre	Année
Impitoyable	1992

Requêtes SQL: SELECT

- Les conditions peuvent être combinées avec «**AND**» **et** «**OR**»

```
SELECT * FROM Film
WHERE (Année =1992
OR Id_realisateur=29)
```

Titre	Année	Id_realisateur	Genre
Impitoyable	1992	20	western
Van Gogh	1990	29	drame

Requêtes SQL: ORDER BY

- On peut trier le résultat d'une requête avec la clause ORDER BY
  - Suivie de la liste des attributs servant de critères de tri.
  - Par défaut le tri se fait par ordre ascendant.
- On peur rajouter le mot-clé DESC après la liste des attributs
  - Pour demander un tri en ordre descendant.

Requêtes SQL: ORDER BY

Exemple

SELECT Titre, Genre **FROM** Film **ORDER BY** Genre, Année

Titre	Genre
Impitoyable	Western
Kagemusha	Drame
Les pleins pouvoirs	Policier
Van Gogh	Drame

Requêtes SQL: clause DISTINCT

- L'utilisation d'une clés permet d'éviter les doublons dans les tables stockées, mais il peuvent apparaître dans le résultat d'une requête. La clause DISTINCT, placé e après SELECT, permet de supprimer ces doublons.
- Exemple:

SELECT Année_Naissance FROM Artiste;

Année_Naissance
1930
1930
1925
Null
1910

Requêtes SQL: clause DISTINCT

- Exemple:

SELECT DISTINCT Année_Naissance FROM Artiste;

Année_Naissance
1930
1925
Null
1910

Requêtes SQL: clause LIKE

On cherche tous les artistes dont la 2ème lettre du nom est un 'a'

- SELECT * FROM Artiste WHERE nom LIKE '_a%'

Id	Nom	Prénom	Année_Naissance
20	Eastwood	Clint	1930
21	Hackman	Gene	1930

- Le caractère '_' désigne n'importe quel caractère;
- Le '%' désigne n'importe quelle chaîne caractères;

Requêtes SQL: clause LIMIT

- Facultative
- A placer à la fin de la requête
- Indique le nombre maximal de lignes dans le résultat.

SELECT * FROM Film LIMIT 3

Titre	Année	Id_realisateur	Genre
Impitoyable	1992	20	western
Van Gogh	1990	29	drame
Kagemusha	1980	68	Drame

Requêtes SQL: clause LIMIT

- On utiliser deux chiffres
- Le premier indique le numéro de la ligne à partir de laquelle la limite s 'applique
  - les lignes sont numérotées à partir de 0

```
SELECT * FROM Film LIMIT 1,3
```

Titre	Année	Id_realisateur	Genre
Van Gogh	1990	29	drame
Kagemusha	1980	68	Drame
Les pleins pouvoirs	1997	20	policier

Requêtes SQL: clause INSERT INTO

- Exemple:

INSERT INTO Film VALUES ('Vertigo', 1958, NULL,'Suspense');

- Si on veut insérer dans une partie seulement des attributs, il faut donner la liste implicitement.
- Exemple:

INSERT INTO Film (titre, année) VALUES ('Vertigo', 1958);

Requêtes SQL: clause DELETE

- Pour être sûr de supprimer le bon enregistrement, utiliser sa clé primaire.
- Exemple:

```
DELETE FROM Film WHERE Id_realisateur= 20
```

Il faut toujours s'assurer que l'on va détruire ce que l'on souhaite!!!

Requêtes SQL: clause UPDATE… SET

- Exemple:

UPDATE Film SET Id_realisateur =20

- Tous les films sont maintenant réalisés par le réalisateur numéro «20»
- UPDATE Film SET Id_realisateur =20 WHERE Année=1992

PHP et les bases de données

PHP/MySQL

PHP/MySQL

- *mysql_pconnect (serveur, nom, passe)*

Etablit une connexion. Si OK, renvoie un identifiant non nul cnx

- *mysql_select_db(base, cnx)*

Se place dans une base et renvoie vrai si OK

PHP/MySQL

- *mysql_query (requete, cnx)*

exécute une requête et renvoie un identifiant

- *mysql_fetch_object (resultat)*

renvoie la ligne suivante sous forme d'objet

PHP/MySQL

mysql_pconnect

Première fonction à appeler pour accéder à MySQL.

- On lui passe le nom du serveur, le login utilisateur et le

mot de passe ;

  - la fonction essaie de se connecter au serveur.
- Si la connexion réussit: un identifiant de connexion est

renvoyé; *sinon la fonction renvoie 0;*

PHP/MySQL
mysql_pconnect

- L'identifiant doit ensuite être utilisé pour tous les appels à MySQL

- Essentiel: bien tester la valeur retournée. Si elle vaut 0, il y a un problème.

PHP/MySQL
mysql_select_db

Seconde fonction: elle sert à choisir la base sur laquelle on travaille.

- On lui passe le nom de la base, et l'identifiant de connexion.
- Le serveur place connecte dans la base choisie, sauf si l'utilisateur n'a pas les droits.

PHP/MySQL
mysql_select_db

- La fonction renvoie o si la connexion à la base échoue, ou un nombre non nul sinon.

- Noter: connexion à MySQL en deux étapes, le serveur puis la base.

PHP/MySQL

Exemple

```
$cnx = mysql_pconnect ("localhost", “Yakouben", "toto");
if ($cnx == 0) {
echo "Connexion à localhost impossible";
exit;
}
if (mysql_select_db ("mabase", $cnx) == 0)
{
echo "Accès à " . BASE . " impossible\n";
exit;
}
```

Mieux vaut isoler cette partie de code dans une fonction.

PHP/MySQL
Exemple de fonction de connexion

```
function Connexion ($nom, $passe, $base, $serveur)
{ // Connexion au serveur
$cnx = mysql_pconnect ($serveur, $nom, $passe);
if ($cnx == 0) {
echo "Connexion à $serveur impossible\n";
exit;
} // Connexion à la base
if (mysql_select_db ($base, $cnx) == 0) {
echo "Accès à $base impossible\n";
echo mysql_error($cnx);
exit;
} // On renvoie la variable de connexion
return $cnx;
}
```

PHP/MySQL
mysql_query

Elle permet d'exécuter n'importe quelle commande SQL.

- On lui passe le texte de la commande SQL, et l'identifiant de connexion.
- Le serveur exécute la commande, et renvoie un *identifiant de résultat*.

PHP/MySQL
mysql_query

- La fonction renvoie o si la commande échoue.
- Noter: la fonction *mysql_error permet de récupérer le texte* du message d'erreur au besoin.

PHP/MySQL
mysql_fetch_object

Elle ramène une ligne d'un résultat d'une *interrogation* SQL. Voici le principe (dit de « curseur »).

- On exécute la requête SELECT avec *mysql_query. On* récupère l'identifiant de résultat.
- Le premier appel à *mysql_fetch_object: renvoie la* première ligne du résultat sous forme *d'objet*.

PHP/MySQL
mysql_fetch_object

- Chaque nouvel appel renvoie la ligne suivante.
- Si la fonction renvoie 0, c'est terminé.

On boucle jusqu'à ce que *mysql_fecth_object renvoie 0.*

PHP/MySQL

Exemple

```
$resultat =
mysql_query ("SELECT * FROM FilmSimple", $connexion);
if ($resultat != 0) {
while($film = mysql_fetch_object ($resultat))
   {
echo "$film->titre, “ . "paru en $film->annee, "
. "réalisé par $film->nomMES.<BR>";
   }
}
else
echo "Erreur rencontrée: ". mysql_error($connexion);
```

PHP/MySQL

Méthodes POST et GET

Utilisation dans un formulaire, par exemple:

- <FORM ACTION="ExMyPHP2.php" METHOD=''POST''>

Ce formulaire vous permet d'indiquer des

paramètres pour la recherche de films :

<P>

Titre : <INPUT TYPE=TEXT SIZE=20 NAME='titre' VALUE='%'>

<P> Année début : <INPUT TYPE=TEXT SIZE=4 NAME='anMin'
VALUE=1900>

Année fin : <INPUT TYPE=TEXT SIZE=4 NAME='anMax' VALUE=2100>

<P>

<INPUT TYPE=SUBMIT VALUE='Rechercher'>

</FORM>

PHP/MySQL

POST

- Spécifie que les données du formulaire seront soumises à la page web “ExMyPHP2.php”
- PHP emmagasine toutes les valeurs "postées" dans un tableau associé
  - Nommé "$_POST".

PHP/MySQL

POST

- Les noms du formulaire sont utilisés en tant que **clés dans le tableau associé** "$_POST »
- Jamais deux entrées d'un formulaire avec le même nom.
- ExMyPHP2.php va contenir

```
<?php
   $titre= $_POST[‘titre'];
   $anMin= $_POST[‘anMin'];
   $anMax= $_POST[‘anMax'];
?>
```

PHP/MySQL

GET (Exemple d'utilisation)

<FORM ACTION="ExMyPHP2.php" METHOD="'GET"'>

Ce formulaire vous permet d'indiquer des paramètres pour la recherche de films :

<P>

Titre : <INPUT TYPE=TEXT SIZE=20 NAME='titre' VALUE='%'>

<P> Année début : <INPUT TYPE=TEXT SIZE=4 NAME='anMin' VALUE=1900>

Année fin : <INPUT TYPE=TEXT SIZE=4 NAME='anMax' VALUE=2100>

<P>

<INPUT TYPE=SUBMIT VALUE='Rechercher'>

</FORM>

PHP/MySQL

GET

- GET passe les variables à la page web " ExMyPHP2.php " en les ajoutant à la fin de l'URL.
- L'URL sera:
  "? titre=xxx&anMin=xxx&anMax' =xxx"
- Le point d'interrogation "?" dit au navigateur que les **objets suivants sont des variables**.

PHP/MySQL

GET

- D'où ExMyPHP2.php va contenir:

```
<?php
…
   $titre= $_GET['titre'];
   $anMin= $_GET['anMin'];
    $anMax= $_GET['anMax'];
…
   ?>
```

Liens

- PHP*
  - http://www.lamsade.dauphine.fr/rigaux/ (Prof P. Rigaux Paris Dauphine)
  - http://www.php.net
  - http://www.phpinfo.net
  - http://www.phpfrance.com
  - http://www.developpez.com/php/
  - http://www.nawouak.net
- MySQL
  - http://www.mysql.com/
  - http://dev.nexen.net/docs/mysql/
- HTML
  - http://cyberzoide.developpez.com/html/
- Exemple concret
  - http://www.miag-rezo.net

- ** PHP Hypertext Preprocessor (encore un acronyme récursif !)*