
Web et Bases de données

- ✓ **Gestion des formulaires**
- ✓ **Les classes**
- ✓ **Accès aux bases de données**
- ✓ **Traitement de sessions**

Sadok Ben Yahia, PhD
Faculté des Sciences de Tunis
Sadok.benyahia@fst.rnu.tn
<http://www.cck.rnu.tn/sbenyahia>



Gestion des formulaires



Formulaires HTML (rappels)

- Exemple :

- ```
<form action="prog.php" method="get">
 <input type="submit" name="go">
</form>
```

- Action : URL cible

- Method : dans la pratique GET ou POST

# Entrées des formulaires HTML (1)

- Boutons d'action

```
<input type="submit" name="nom" value="valeur">
```

```
<input type="reset" name="nom" value="valeur">
```

```
<input type="image" name="nom" src="source">
```

- Entrées textuelles

```
<input type="text" name="nom" value="valeur">
```

```
<input type="hidden" name="nom" value="valeur">
```

```
<input type="password" name="nom" value="valeur">
```

- Boîtes de saisie

```
<textarea name="nom" rows="r" cols="c">
```

```
 texte...
```

```
</textarea>
```

## Entrées des formulaires HTML (2)

- Listes à choix

```
<select [multiple] name="nom">
 <option value="valeur">texte</option>
 ...
</select>
```

- Boutons radio

```
<input type="radio" name="nom"
 value="valeur" [selected] >
```

- Boutons à cocher

```
<input type="checkbox" name="nom"
 value="valeur" [checked] >
```

# Vérification des Paramètres

```
<form method="post"
 action="verif.php">
 <input type="text" name="nom">
 <input type="text" name="prenom">
 <input type="submit"
 name="bouton" value="Envoyer">
</form>
```

```
<?
if(empty($nom) ||
empty($prenom))//on vérifie avec
empty voir si les champs sont vide
{
print "le champ nom ou le champ
prenom est vide"; //si un des 2
champs n'est pas rempli, message
d'erreur
}
else{ //sinon message de
confirmation
print "les champs sont ok";
}
?>
```

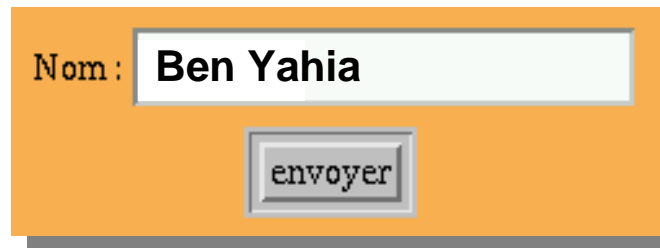
# Paramètres CGI (1)

- Sont des variables globales
- Quelque soit la méthode CGI utilisée
  - `HTTP_POST_VARS`, `HTTP_GET_VARS`
- Peuvent être des tableaux

## Paramètres CGI (2)

```
<form action="exemple.php" method="get">
Nom : <input type="text" name="nom">

<input type="submit" name="bouton"
 value="envoyer">
</form>
```



Nom : Ben Yahia

envoyer

- Dans exemple.php
  - `$GLOBALS["bouton"]` vaut "envoyer"
  - `$GLOBALS["nom"]` vaut ce qu'il y a dans le dialogue associé



## Paramètres CGI (3)

```
<form action="tab.php" method="get">
```

```
Nom : <input type="text" name="p[nom]">

```

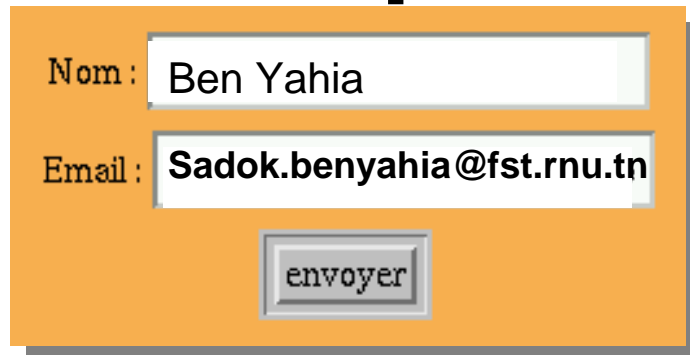
```
Email : <input type="text" name="p[email]">

```

```
<input type="submit" name="bouton"
```

```
value="envoyer">
```

```
</form>
```



- Dans **tab.php**

```
$GLOBALS["bouton"] vaut "envoyer"
```

```
$GLOBALS["p"]["nom"] et $GLOBALS["p"]["email"] valent ce qu'il y a dans les dialogues associés
```

## Paramètres CGI (4)



```
<form action="selection.php" method="get">
 <select multiple name="selection[]">
 <option value="x">Mlle X</option>
 <option value="y">Mme Y</option>
 <option value="z">M. Z</option>
 </select>
 <input type="submit" name="bouton"
 value="envoyer">
</form>
```

- Dans selection.php
  - `$GLOBALS["bouton"]` vaut "envoyer"
  - `$GLOBALS["selection"]` a pour taille le nombre d'options sélectionnées et pour valeurs les options sélectionnées

## Paramètres CGI (5)

```
<form action="image.php" method="post">
Cliquez sur l'image :

<input type="image" src="image.gif"
name="bout">
</form>
```



- Dans image.php
  - `$GLOBALS["bout_x"]` et `$GLOBALS["bout_y"]` ont pour valeur les coordonnées du point de l'image sur lequel l'utilisateur a cliqué

## Exemple

```
<form action="test_cgi.php" method="GET">
 Texte :

 <input type="text" name="texte_court" value="blabla"><hr>
 Sélection simple :

 <select name="sel_simple">
 <option value="1">Choix 1</option>
 <option value="2">Choix 2</option>
 <option value="3">Choix 3</option>
 </select><hr>
 Sélection multiple :

 <select multiple name="sel_multiple[]">
 <option value="1">Choix 1</option>
 <option value="2">Choix 2</option>
 <option value="3">Choix 3</option>
 </select><hr>
 <input type="reset" name="bouton_reset" value="Annuler">
 <input type="submit" name="bouton_submit"
value="Valider">
</form>
```

# Exemple

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

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

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

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

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

Test CGI - Microsoft...  
Fichier Edition Affic| >>  
Test CGI  
Texte :  
blabla  
Sélection simple :  
Choix 2  
Sélection multiple :  
Choix 1  
Choix 2  
Choix 3  
Annuler Valider  
Intranet local

`$texte_court : "blabla"`

`$sel_simple : "2"`

`$sel_multiple[0] : "1"`  
`$sel_multiple[1] : "3"`

`$bouton_submit : "Valider"`

---

# Les classes

---



# Classes (1)

- Une classe est un ensemble de variables et de fonctions travaillant sur ces variables
- Un objet est une instance particulière d'une classe
- POO de base, mais, grâce à l'héritage
  - réutilisabilité du code
  - modularité

## Classes (2)

- Possibilité de surcharger des méthodes (fonctions) d'une classe
- Attention !
  - pas d'appel automatique des constructeurs des classes ancêtres !
  - Pas de destructeur
- Dans une méthode, `$this` désigne l'objet dont on exécute la méthode



## Classes (définition)

- Exemple : la classe `exemple` maintient un entier `n` affiché et incrémenté à chaque appel de sa méthode `plus()`.

```
class exemple {

 var $n ; // attributs

 function exemple($init=1) //
 constructeur
 { $this->n = $init ; }

 function plus() // méthode
 { echo "$this->n
" ; $this->n ++ ; }

} // fin de la classe exemple
```

## Classes (héritage)

- Exemple : la classe `sur_exemple` possède une méthode supplémentaire `moins()` qui décrémente l'attribut

```
class sur_exemple extends exemple { // héritage

function sur_exemple($init=1) // constructeur
{ $this->exemple($init) ; }
// ! pas d'appel automatique

function moins() // méthode
{ echo "$this->n
" ; $this->n -- ; }

} // fin de la classe sur_exemple
```

- Pas d'héritage multiple

## Classe (instanciation)

<?

```
$e = new sur_exemple(4) ;
$e->plus() ; // 4
$e->moins() ; // 5
$e->plus() ; // 4
```

?>

---

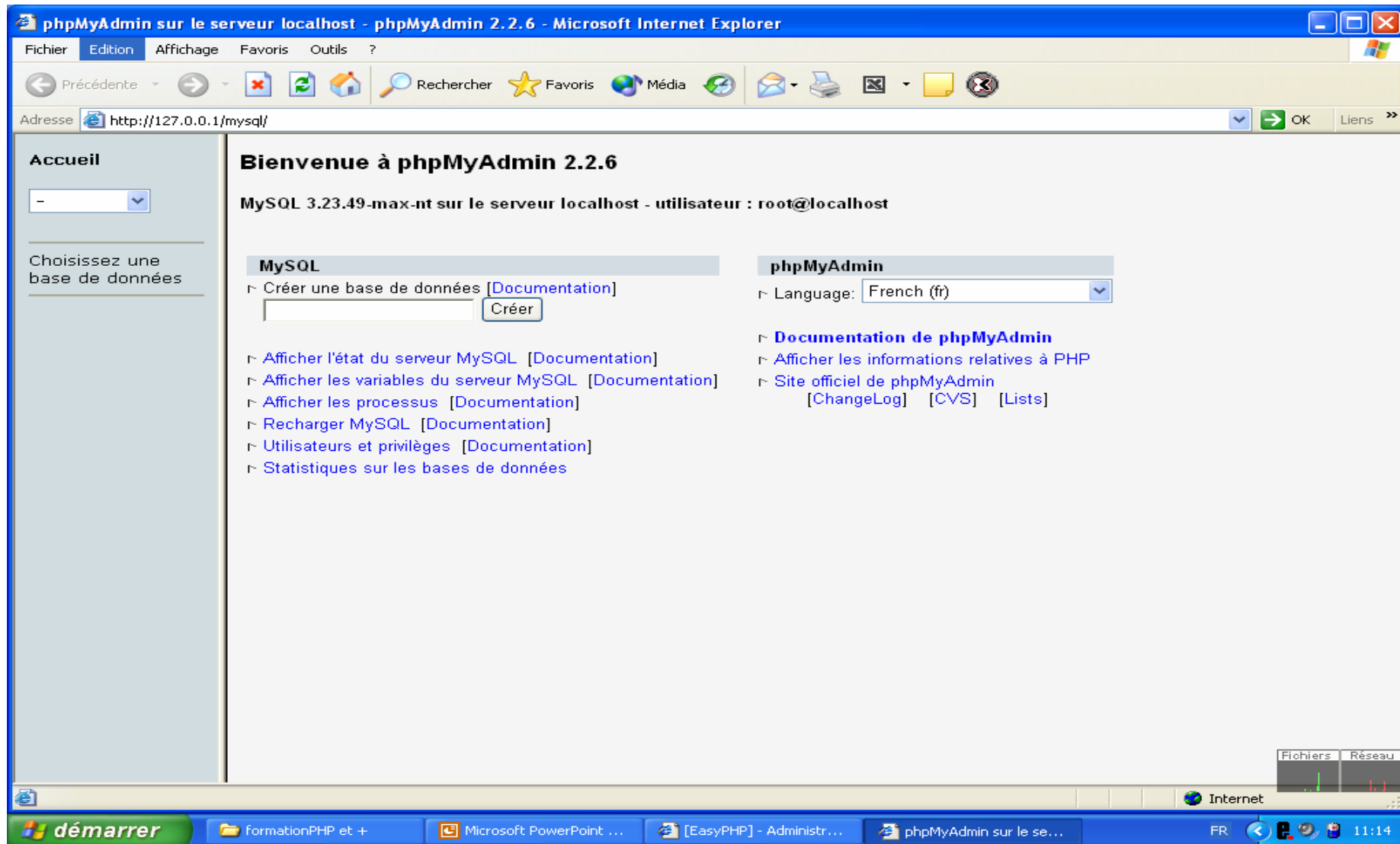
# Accès aux bases de données

---



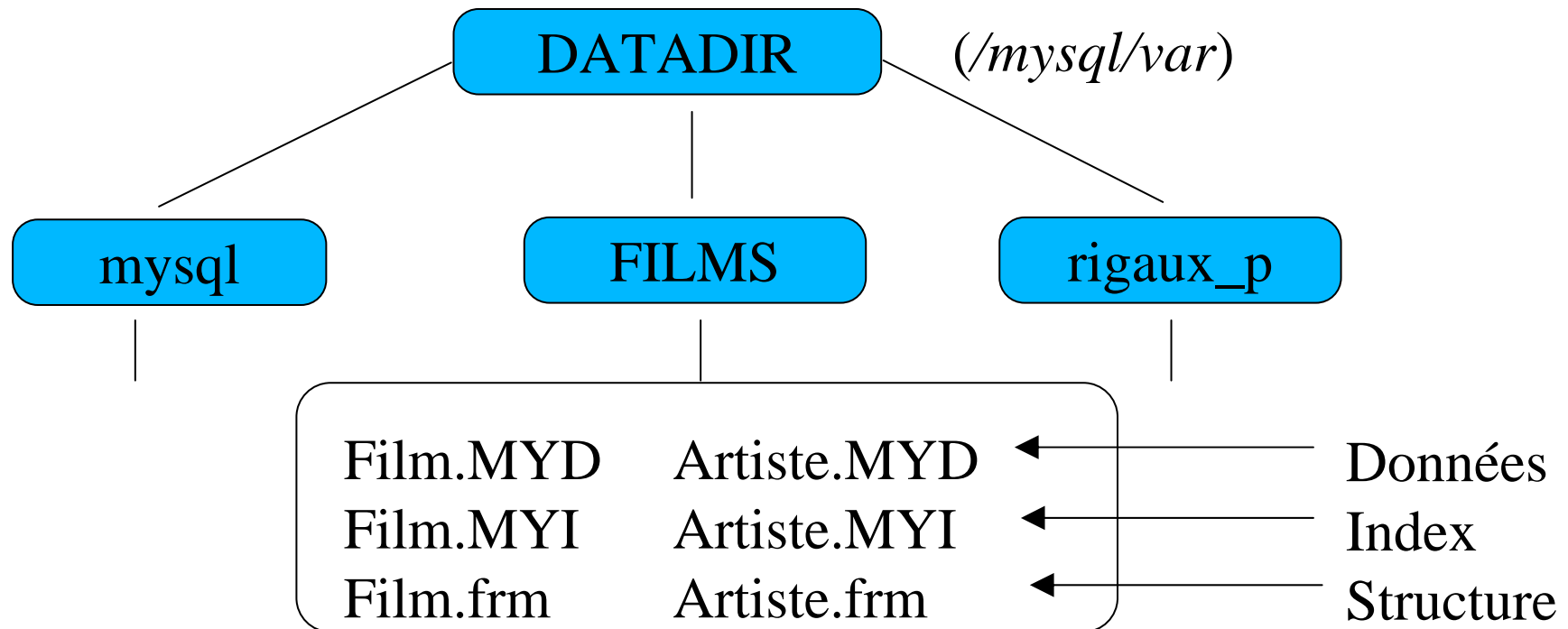
# PHP et les bases de données

- Un support très large
  - Adabas D, dBase, filePro, Hyperwave, Informix, InterBase, mSQL, MS SQL Server, MySQL, Oracle 7, Oracle 8, SyBase, PostgreSQL, Solid, ODBC
- Utiliser la bibliothèque de son SGBD
- En l'absence de bibliothèque, utiliser ODBC
  - ou développer sa propre bibliothèque



# Les bases de données

- Une base = un répertoire !
- Une table = 3 fichiers !



# Emplacement de stockage de la base de données

[EasyPHP] - Administration - Microsoft Internet Explorer

Fichier Edition Affichage Favoris Outils ?

Précédente Recherche Favoris Média

Adresse <http://127.0.0.1/home/> OK Liens

**ADMINISTRATION**

- Vos alias : [ajouter]
- Données MySQL (datadir) :** répertoire actuel : "C:\Program Files\EasyPHP\mysql\data" [modifier]
- Administrez vos bases de données :**  
Cliquez sur le bouton ci-dessous pour accéder à l'administration des bases de données.  
**"PhpMyAdmin"**
- Environnement EasyPHP :**  
Ces pages vous informeront sur le bon fonctionnement de PHP, sa configuration et sur les éléments installés.  
**infos php extensions paramètres retour**

Bienvenue dans votre environnement EasyPHP.  
Si vous voyez cette phrase, PHP fonctionne. Pour vérifier le fonctionnement de MySQL, vous pouvez accéder au centre d'administration : "PhpMyAdmin".  
Si vous rencontrez des problèmes, reportez vous au site d'EasyPHP : [www.easyphp.org](http://www.easyphp.org)

Terminé Internet

démarrer Boîte de réception ... formationPHP et + php\_Formation\_IN... php\_Formation\_IN... [EasyPHP] - Admini... FR 15:10



# Types de données MySQL

- MySQL reconnaît les principaux types SQL2, et quelques autres:
  - INTEGER : les entiers
  - FLOAT : numériques flottants
  - DECIMAL (M, D) : numériques exacts
  - CHAR : chaînes de longueur fixe
  - VARCHAR : chaînes de longueur variable
  - TEXT : textes longs
  - DATE : dates
  - TIME : moments

# Exemple de création de table

```
CREATE TABLE Film
```

```
(id INT NOT NULL AUTO_INCREMENT
titre VARCHAR (50) NOT NULL,
annee INTEGER NOT NULL,
idMES INTEGER,
resume TEXT,
codePays VARCHAR (4),
PRIMARY KEY (titre),
FOREIGN KEY (idMES) REFERENCES
```

```
Artiste,
```

```
FOREIGN KEY (codePays) REFERENCES
```

```
Pays);
```

## Insertion dans une table

```
mysql> CREATE TABLE MaTable
-> (MaDonnee INTEGER NOT NULL,
-> PRIMARY KEY (MaDonnee));
Query OK, 0 rows affected (0.02 sec)
```

```
mysql> INSERT INTO MaTable (MaDonnee) VALUES (1);
Query OK, 1 row affected (0.05 sec)
```

- Il vaut mieux placer toutes les commandes dans un fichier de script, *MonF.sql*  
*mysql -u MonNom -p MaBase < MonF.sql*

## Exemple d'un fichier de script

```
Création d'une table 'FilmSimple'
```

```
CREATE TABLE FilmSimple
 (titre VARCHAR (30),
 annee INTEGER,
 nomMES VARCHAR (30),
 prenomMES VARCHAR (30),
 anneeNaiss INTEGER
)
;
```

# MySQL en résumé

- Tout à fait conforme à la norme SQL ANSI
  - CREATE TABLE, DROP TABLE, ALTER TABLE
  - SELECT, INSERT, DELETE, UPDATE
- Avec des extensions (très utiles)
  - Types de données (TEXT, BLOB)
  - Contraintes (AUTO\_INCREMENT)
  - Beaucoup de fonctions

## Connexions persistantes (aux bases de données)

- La connexion à une base de données est souvent l'opération la plus coûteuse d'une requête
- Identification d'une connexion :  
(*host, database, user, password*)
- Au lieu d'ouvrir et fermer une connexion à une base de données à chaque requête, on peut garder les connexions ouvertes et les mémoriser pour pouvoir les réutiliser par la suite

## Accès à un SGBD (schéma)

- Connexion ([p]=persistante)

```
$conn = xxx_[p]connect(serveur,
 port,
 base,
 utilisateur,
 mot_de_passe) ;
```

- Interrogation/mise à jour

```
$query = xxx_query(requête) ;
$res = xxx_fetch($query) ;
```

- Fermeture

```
xxx_close($conn) ;
```

## ■ Connexion

```
$conn = mysql_[p]connect(serveur,
 utilisateur,
 mot_de_passe) ;
$bdd = mysql_select_db(base,$conn) ;
```

## ■ Interrogation/mise à jour

```
$query = mysql_query(requête) ;
$res = mysql_fetch_[row|array|assoc]($query) ;
```

## ■ Fermeture

```
mysql_close($conn) ;
```



---

## Programmation MySQL/PHP : premier exemple

- ✓ Connexion à la base MaBase (base de films)
- ✓ Affichage du titre, année et metteur en scene du film

## Exemple : interrogation et affichage (1)

```
<?php
$connexion = mysql_pconnect ("localhost", "MonNom",
 "MonMDP");

if (!$connexion)
{
 echo "Désolé, connexion impossible\n";
 exit;
}

if (!mysql_select_db ("MaBase", $connexion))
{
 echo "Désolé, accès à la base impossible\n";
 exit;
}
```

## Exemple : interrogation et affichage (2)

```
$resultat = mysql_query ("SELECT * FROM FilmSimple",
 $connexion);

if ($resultat)
 while ($film = mysql_fetch_object ($resultat))
 echo "$film->titre, paru en $film->annee, réalisé "
 . "par $film->prenomMES $film->nomMES.
\n";
else
{
 echo "Erreur dans l'exécution de la
requête.
";
 echo "Message : " .mysql_error($connexion);
```

---

# Programmation MySQL/PHP : exemple avec formulaire

# En association avec un formulaire

```
<H1>Formulaire pour programme PHP</H1>
```

```
<FORM ACTION='ExMyPHP2.php' METHOD=POST>
```

```
<P>
```

```
Titre : <INPUT TYPE=TEXT SIZE=20 NAME = 'titre'> <P>
```

```
Année début : <INPUT TYPE=TEXT SIZE=4 NAME='anMin'>
```

```
Année fin :<INPUT TYPE=TEXT SIZE=4 NAME='anMax'> <P>
```

```
<P>
```

```
Comment combiner ces critères.
```

```
ET <INPUT TYPE=RADIO NAME='comb' VALUE='ET'>
```

```
OU <INPUT TYPE=RADIO NAME='comb' VALUE='OU'> ?
```

```
<INPUT TYPE=SUBMIT VALUE='Rechercher'>
```

```
</FORM>
```

## On récupère donc les variables \$titre, \$anMin, \$anMax, et \$comb

```
<?php
 echo "Titre = $titre anMin = $anMin anMax=
 $anMax\n";
 echo "Combinaison logique : $comb<P>\n";

 if ($comb == 'ET')
 $requete = "SELECT * FROM FilmSimple "
 . "WHERE titre LIKE '$titre' "
 . "AND annee BETWEEN $anMin and $anMax";
 else
 $requete = "SELECT * FROM FilmSimple "
 . "WHERE titre LIKE '$titre' "
 . "OR (annee BETWEEN $anMin and $anMax)";
```

## Il reste à exécuter la requête

```
<?php
```

```
 $connexion = mysql_pconnect (SERVEUR, NOM, PASSE);
```

```
mysql_select_db (BASE, $connexion);
```

```
 $resultat = mysql_query ($requete, $connexion);
```

```
 while (($film = mysql_fetch_object ($resultat)))
```

```
 echo "$film->titre, paru en $film->annee,
```

```
réalisé "
```

```
 . "par $film->prenomMES $film-
```

```
>nomMES.
\n";
```

```
?>
```

---

# Programmation MySQL/PHP : mise à jour d'une base



## Mises à jour de la base

- On utilise un formulaire de saisie, et on déclenche:
  - Un ordre INSERT pour des insertions
  - Un ordre UPDATE pour une modification
  - Un ordre DELETE pour une destruction
- Dans tous les cas la fonction *mysql\_query* permet d'exécuter l'ordre.

# Exemple: mise à jour de la table FilmSimple

```
<FORM ACTION="ExMyPHP3.php" METHOD=POST>
```

```
Titre : <INPUT TYPE=TEXT SIZE=20 NAME="titre">

```

```
Année : <INPUT TYPE=TEXT SIZE=4 NAME="annee"> <P>
```

```
Nom : <INPUT TYPE=TEXT SIZE=20 NAME="prenom">

```

```
Prénom : <INPUT TYPE=TEXT SIZE=20 NAME="nom">

```

```
Année : <INPUT TYPE=TEXT SIZE=4 NAME="anneeNaissance">
```

```
<H1>Votre choix</H1>
```

```
<INPUT TYPE=SUBMIT VALUE='Insérer' NAME='insérer' >
```

```
<INPUT TYPE=SUBMIT VALUE='Modifier' NAME='modifier' >
```

```
<INPUT TYPE=SUBMIT VALUE='Détruire' NAME='detruire' >
```

```
</FORM>
```

# L'action depend du bouton choisi par l'utilisateur

```
// Test du type de mise à jour effectuée
```

```
echo "<HR><H2>\n";
if (isset($inserer)) echo "Insertion du film
$titre";
elseif (isset($modifier)) echo "Modification du film
$titre";
elseif (isset($destruire)) echo "Destruction du film
$titre";
echo "</H2><HR>\n";
```

```
// Affichage des données du formulaire
```

```
echo "Titre: $titre
 annee: $annee
\n";
echo "Mis en scène par $prenom $nom, né en
$anneeNaiss\n";
```

## Choix de la requête en fonction de l'action demandée

```
if (isset($inserer))
 $requete = "INSERT INTO FilmSimple (titre, annee, "
 . "prenomMES, nomMES, anneeNaiss) "
 . "VALUES ('$titre', $annee, "
 . "'$nom', '$prenom', $anneeNaiss) ";
if (isset($modifier))
 $requete = "UPDATE FilmSimple SET annee=$annee, "
 . "prenomMES = '$prenom', nomMES='$nom', "
 . "anneeNaiss=$anneeNaiss WHERE titre =
'$titre' ";
if (isset($detruire))
 $requete = "DELETE FROM FilmSimple WHERE
titre='$titre'";

$resultat = mysql_query ($requete, $connexion);
```

---

# Programmation MySQL/PHP : requêtes sur la structure de la base

## Informations sur le schéma

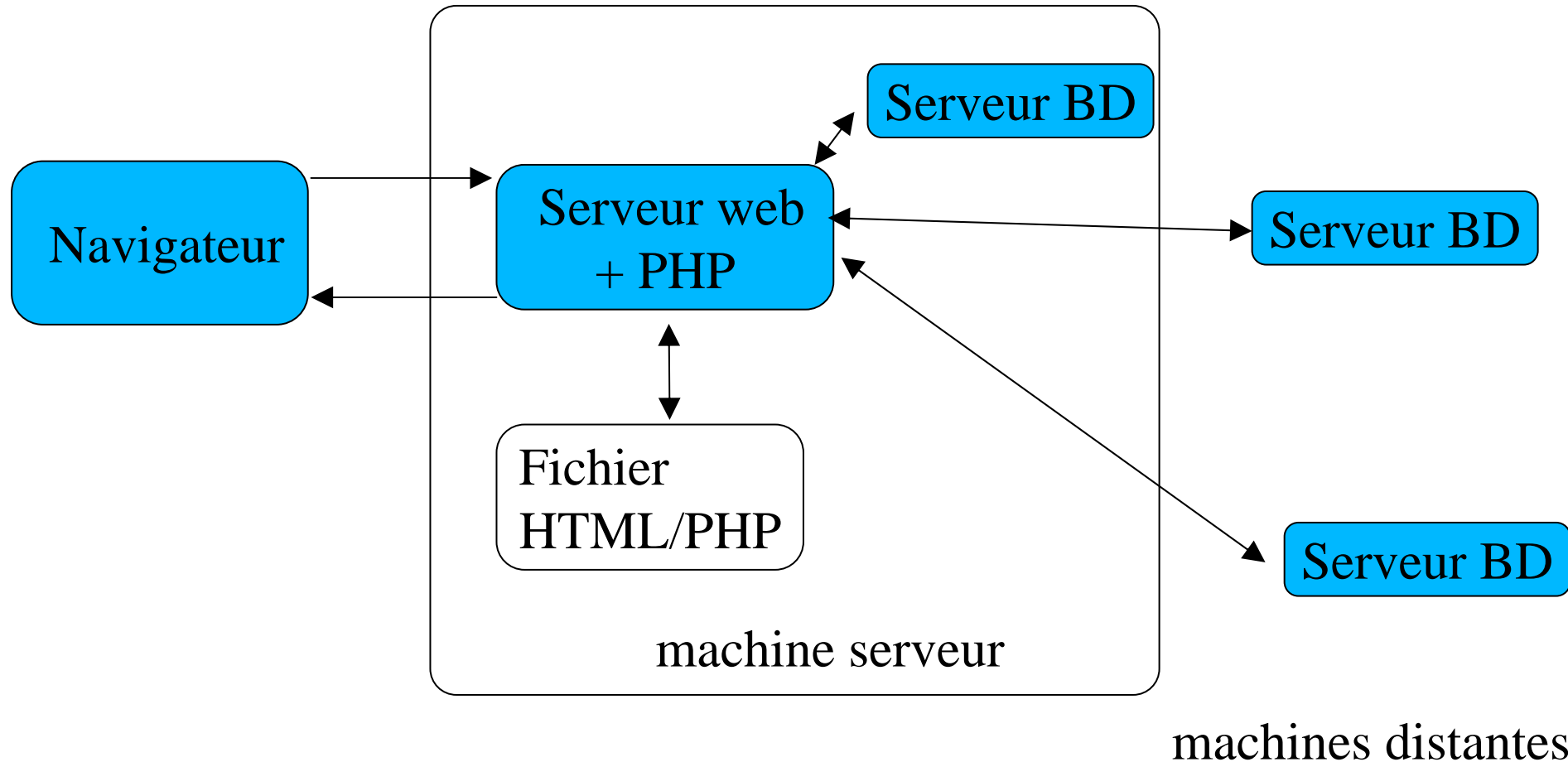
- Etant donné le résultat d'une requête :
  - `mysql_num_fields($res)` donne le nombre d'attributs
  - `mysql_field_name($res, $i)` donne le nom de l'attribut
  - `mysql_fetch_row($res)` renvoie une ligne du résultat sous la forme d'un tableau indicé.

---

## Connexions :

- A plusieurs bases
- A plusieurs SGBD

# Interroger plusieurs bases Mysql





## Plusieurs connexion php

```
<?php
```

```
$connexion1 = mysql_pconnect (serveur1, nom1, pass1);
mysql_select_db (base1, $connexion1);
```

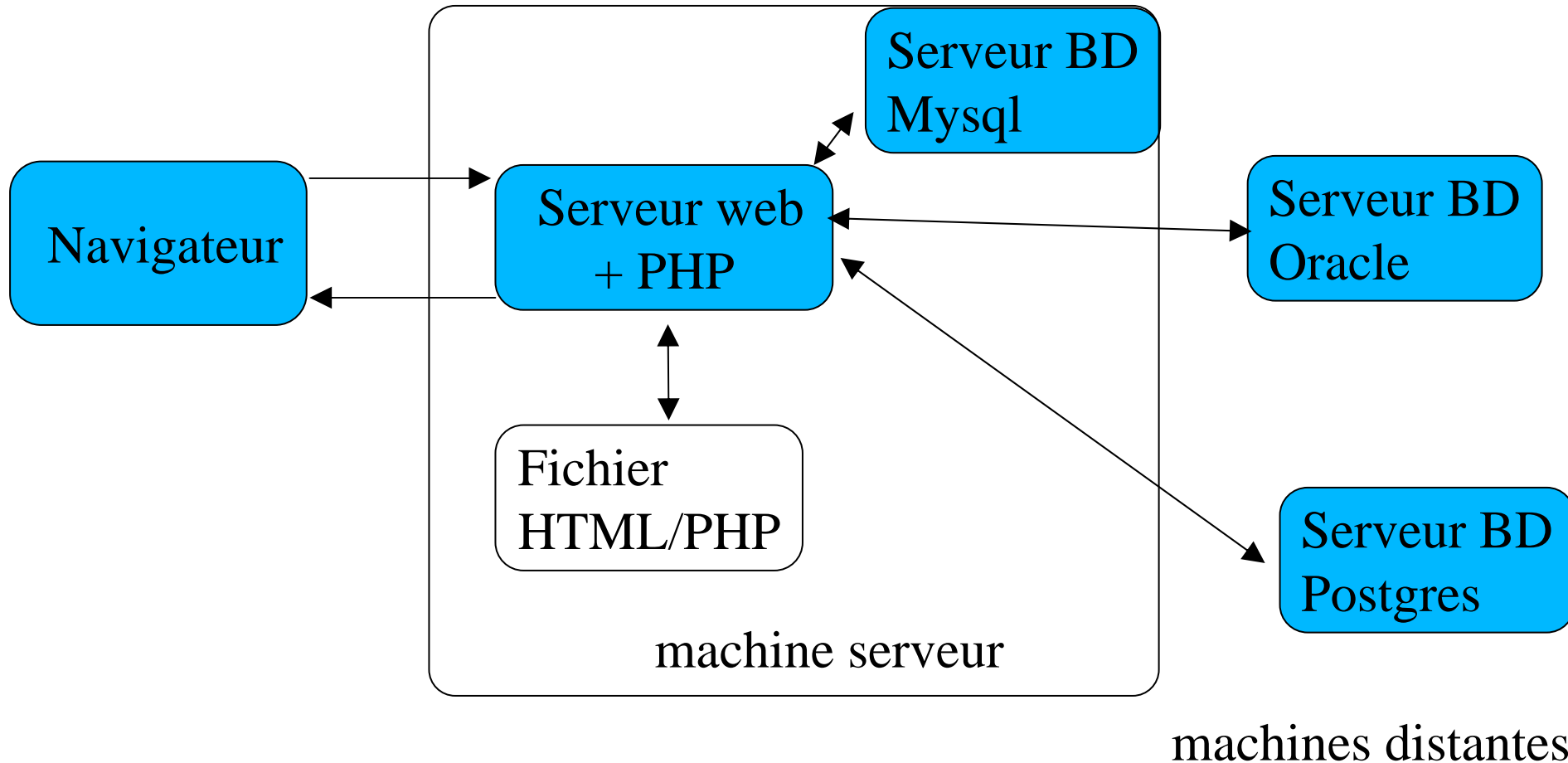
```
$connexion2 = mysql_pconnect (serveur2, nom2, pass2);
mysql_select_db (base2, $connexion2);
```

```
$resultat1 = mysql_query ($requete1, $connexion1);
$resultat2 = mysql_query ($requete2, $connexion2);
```

```
traitement($resultat1,$resultat2);
```

```
?>
```

# Et si la base n'est pas Mysql ?



---

## Interface commune

- ODBC (Open Database Connectivity)
- Fonctions php valable pour tout SGBD
  
- `odbc_connect(db,login,pass);`
- `odbc_exec(requete,connexion);`
- `odbc_fetch_row(connexion);`

## Descripteur (aperçu)

```
<?php
```

```
 $connexion1 = odbc_connect ("mysql:localhost",...);
```

```
 $connexion2 = odbc_connect ("oracle:db.fournisseur.fr" ...);
```

```
 $connexion3 = odbc_connect ("postgres:128.176.212.3",...);
```

```
?>
```

## PHP & ODBC (Access, SQL Server, Oracle, ...)

<Permettre à une page PHP d'envoyer des requêtes SQL à une base ODBC

⇒ librairies de fonctions prédéfinies

### 1. Ouverture d'une connexion

```
$cx = odbc_connect(nom de la source ODBC , login , mot de passe);
```

```
$cx = odbc_connect("EtatCivil", "", "");
```

### 2. Envoi d'une requête SQL pour la base EtatCivil

```
$result = odbc_exec(connexion , requête SQL);
```

```
$result = odbc_exec($cx, "SELECT * FROM ages");
```

### 3. Récupération du résultat tuple à tuple

```
</ body> </ html>
```

```
while (odbc_fetch_row($ result)) {
```

```
echo odbc_result($ result, 1); # le nom
```

```
echo odbc_result($ result, 2); # le prenom
```

### 4. Libération des données résultat et fermeture de la connexion

```
odbc_free_result($ result); odbc_close($cx);
```

- Objectif : Transmettre des variables et leur valeur d'une page web à l'autre

**Méthode 1** : Utiliser les champs cachés d'un formulaire

**Méthode 2** : Ajouter des paramètres à l'URL de la page cible dans un lien hypertexte

**Exemple** : On dispose du nom du visiteur de la page courante dans la variable PHP \$nom et on souhaite le transmettre à la page suite.php3.

## Passage de paramètres via une URL

### Page web courante : courante.php3

```
<!-- Méthode 1 -->
<FORM ACTION="suite.php3" METHOD="POST"
ENCTYPE="multipart/form-data">
<INPUT TYPE="hidden"
NAME="nom"
VALUE="<? echo $nom; ?>" >
<INPUT TYPE="submit" VALUE="Envoyer">
</FORM>
```

### <!-- Méthode 2 -->

```
<A HREF="suite.php3?nom=<? echo $nom; ?>">Envoyer
```

<!-- NB : Il est possible de passer plusieurs paramètres en les séparant par des & -->

### Page web suivante : suite.php3

```
<? /* Récupération de la valeur de la variable */
Uniquement nécessaire avec la méthode 2
parse_str(urldecode($argv[0]));
echo $nom;
?>
```

## Classe générique de connexion

```
function send_sql($sql) {
$res = mysql_query($sql) or die ('
Erreur lors de la tentative
d'envoi d'une requete :
- Requete : '.$sql.'
- Erreur :
' .mysql_error());
return $res;}
```

```
$lk = connect();
Puis pour l'envoi d'une requete de faire ceci :
$sql = "SELECT id, nom, prenom FROM table";
$res = send_sql($sql);
while ($row = mysql_fetch_array($res)) {
echo 'affiche moi les enregistrements';
```



## Classe générique de connexion

```
function connect() {
$host = 'localhost'; // Hôte du server SQL
$login = 'root'; // Votre login
$pwd = ""; // Votre mot de passe
$db = 'base_de_donnee'; // Nom de la base de donnée
$linkid = mysql_connect($host, $login, $pwd) or die ('

Erreur lors de la tentative de connexion au server
MySql :
- Hôte : '.$host.'
- Erreur :
' .mysql_error());
```

```
$sel_db = mysql_select_db($db, $linkid) or die ('

Erreur lors de la tentative de sélection de la base de
donnée :
- Hôte : '.$host.'
- Base de donnée :
' . $db.'
- Erreur : ' .mysql_error());
return $linkid;
```

---

# Traitement des sessions

---



## Problèmes

- Le serveur ne distingue pas les connexions successives : comment identifier un utilisateur ?
- Comment faire pour que les données relatives à un utilisateur soit disponibles pour tous les scripts du site web ?

---

## Qu'est-ce qu'une session ?

- Une session est en quelque sorte l'équivalent du cookie avec quelques différences.
- En effet, une session est un fichier stocké à l'inverse du cookie non pas sur le disque du visiteur mais sur le serveur.
- Une session est personnelle à chaque visiteur (un fichier par visiteur), et permet comme le cookie de stocker temporairement des variables de type primitif.

# Qu'est-ce qu'une session ?

- Les variables de "type primitif" sont les variables chaînes, entières, flottantes, caractères, tableaux, ...
- La durée d'une session est définie par l'hébergeur mais on peut parfois modifier s'il l'autorise la durée par défaut de conservation de la session car elle est automatiquement supprimée en général au bout de 30 minutes.

## A quoi servent-elles ?

- Pister le visiteur (et établir des statistiques précises),
- transporter aisément des variables de page en page,
- elles ont aussi une capacité de stockage plus élevée qu'un cookie.
- Utilisées en complément des cookies dans les scripts d'identification puisque parfois le client n'accepte pas les cookies, et donc dans ce cas la session remplace parfaitement le cookie, stocke son identifiant et son mot de passe crypté pour que le membre n'est pas à le répéter sur chaque page.

## Fonctionnement

- Les sessions permettent de conserver des variables tant qu'il reste connecté au site et environ 30 minutes après
- La session est unique, un identifiant est donc généré aléatoirement pour chaque session et est stocké dans un cookie soit passé dans la barre d'adresse (méthode GET).
- Toutes les sessions dans la plupart des cas sont stockées dans un même répertoire sur le disque du serveur, et une session est débutée manuellement.

# Débuter une session(1)

- automatique sur toutes vos pages si l'option `session.auto_start` est égale à 1. Cette option a pour valeur par défaut 0 (déconseillée).

## Extrait du fichier php.ini (Windows) :

- `session.auto_start = 0 ; initialize session on request startup`



## Débuter une session(2)

- Démarrage explicite et manuel de la session en utilisant la fonction **session\_start()**.
- Elle permet non seulement de créer une session mais aussi de restaurer une session via l'identifiant de session.
- Recommandable pour démarrer les sessions : elle permettra de récupérer la session existante ou d'en créer une si aucune session ne correspond à l'utilisateur.

## Débuter une session(3)

- Débuter implicitement une session en utilisant la fonction **`session_register()`**. Cette fonction permet d'enregistrer une variable dans une session, et si aucun appel à **`session_start()`** n'a été fait avant l'enregistrement direct de variable(s), alors **`session_start()`** sera appelée implicitement.

# Sauvegarder une variable

- `session_register()` : sauvegarde une variable de type chaîne, entier, flottant, caractère, ...
- Elle prend en paramètre sous forme de chaîne le nom de la variable à enregistrer et renvoie un booléen vrai ssi la variable a été enregistrée.

## Exemple :

```
<?php
session_start(); //Création de la session
$prenom = "XXY";
if(session_register("prenom")) { //Sauvegarde dans la session créée
de la variable $prenom
 echo "Variable \$prenom sauvegardée !";
} else {
 echo "Erreur : la variable \$prenom n'a pu être sauvegardée !";
} ?>
```

# Récupération de données dans une session

## ■ Avant PHP 4.1.0 :

On récupère le contenu d'une variable de session en utilisant le tableau associatif suivant :

```
$HTTP_SESSION_VARS["nom_de_votre_variable"]
```

## A partir de PHP 4.1.0 (inclus) :

On récupère le contenu d'une variable de session en utilisant le tableau associatif suivant :

```
$_SESSION["nom_de_votre_variable"]
```

## Exemple :

afficher la valeur de la variable de session *prenom* :

```
<?php
//Version antérieure à 4.1.0
echo $HTTP_SESSION_VARS["prenom"];
```

```
//Version supérieure ou égale à 4.1.0
echo $_SESSION["prenom"]; ?>
```

# Savoir si une variable appartient à une session

- Utiliser la fonction **session\_is\_registered()** : renvoie vrai si le nom de la variable passé en argument est effectivement actuellement présente dans la session.

## Exemple

```
<?php
 if(session_is_registered("prenom")) {
 echo "La variable prenom est déjà enregistrée !";
 } else {
 echo "La variable prenom n'est pas enregistrée !";
 //On la demande puis on la sauvegarde
 } ?>
```

Il est aujourd'hui conseillé d'utiliser **isset()** qui prend en argument la variable à tester

## Exemple :

```
if(isset($_SESSION["prenom"])) {
 echo "La variable prenom est déjà enregistrée !"; }
else {
 echo "La variable prenom n'est pas enregistrée !";
} ?>
```

# Supprimer une variable d'une session

- Pour supprimer une variable, uniquement, d'une session, on utilise la fonction **session\_unregister()**, qui prend en paramètre le nom de la variable à détruire.
- Elle renvoie vrai en cas de succès et bien sûr faux en cas d'erreur.
- Exemple :

```
<?php
 if(session_unregister("prenom")) {
 echo "Variable prenom supprimée avec succès !";
 } else {
 echo "La suppression de la variable prenom a
échoué !";
 }
?>
```

# Détruire une session

- **session\_destroy()** : sans paramètres et qui renvoie vrai en cas de succès et faux en cas d'erreur. A utiliser quand on veut se déconnecter

## Exemple :

```
<?php
 if(session_destroy()) {
 echo "Session détruite !"; }
else {echo "Erreur:impossible de détruire la session ";
 } ?>
```

Il est possible de détruire toutes les variables de session en conservant la session: **session\_unset()**, qui ne prend rien en paramètre et ne retourne aucune valeur.

## Exemple :

```
<?php
 session_unset(); ?>
```

## Exemple session

```
<form action="login.php" method="post">
votre pseudo : <input type="text"
 name="pseudo" value="">votre pwd :
 <input type="password" name="pwd"
 value="">validez : <input type="submit"
 name="go" value="OK">
```

login.php

```
<?php
$pseudo_valide="moi";
$pwd_valide="123l4";
$_SESSION["pseudo_valide"]=$pseudo;
if ($_POST["pseudo"]== $pseudo_valide &&
$_POST["pwd"]== $pwd_valide)
{ header('location: membre.php'); // redirige vers la page
"membre.php" }
else { echo "pseudo ou password invalide :(" ; }
?>
```



## Exemple session

La fonction `session_start()` doit être appelée avant toute balise `html`. Elle doit être présente dans chaque page où l'on fait appel aux variables de session.

**membre.php :**

```
<?php session_start(); // démarre la session?>
```

```
<html>
```

```
<body>
```

```
<?php echo "Bonjour et bienvenue, ".$_SESSION["pseudo_valide"]."
```

```
!";?>etc...
```

```
Déconnexion(<?php echo
$_SESSION["pseudo_valide"] ?>)
```

```
 etc... >
```

---

# Exemple session

fichier deconnect.php :

```
<?php session_start();
```

```
session_unset();// supprime toutes les variables de session
```

```
session_destroy();// deux précautions valent mieux qu'une
```

```
echo "vous devriez à présent être déconnecté(e) :)";?>
```

**Merci**

