# PHP/MySQL: Sites dynamiques
# Formulaires et bases de données

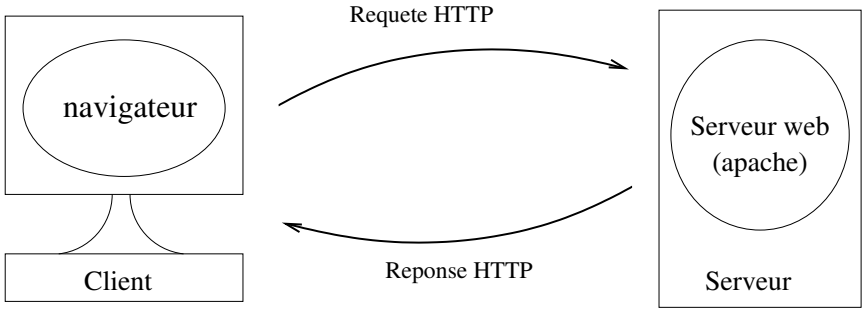
François Gannaz

INP Formation Continue

16, 17, 23, 24 juin 2011



## Web statique : client-serveur






## Discussion HTTP

### Exemple de requête HTTP

GET /page-test.html HTTP/1.1
Host : www.exemple.fr
User-Agent : Opera/9.22 (X11 ; Linux x86_64 ; U ; en)
Accept : text/html, application/xml ;q=0.9
Accept-Charset : iso-8859-1, utf-8

### Exemple de réponse HTTP

HTTP/1.1 200 OK
Date : Tue, 13 Nov 2007 10 :32 :48 GMT
Server : Apache/2.0.52 (CentOS)
Accept-Ranges : bytes
Connection : close
Content-Type : text/html ; charset=UTF-8
*. . . [HTML de la page] . . .*



## Caractéristiques du HTTP

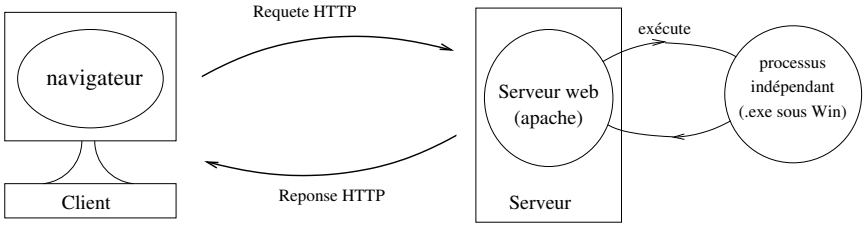
**Un dialogue client-serveur est formé de :**

- Requête HTTP
  - source d'informations variées :
    IP, version du navigateur, OS, langue, page de provenance, etc.
- Réponse HTTP
  - entête + contenu
  - contenu pas forcément en HTML : **Content-Type** (MIME)
  - permet la redirection d'URL

### Exemple de réponse : Redirection HTTP

HTTP/1.x 301 Moved Permanently
Date : Tue, 13 Nov 2007 10 :32 :48 GMT
Location : http ://newserver.fr/page.php

# Serveur : Common Gateway Interface (CGI)




- Le serveur web (apache) lance un nouveau processus
- ce processus envoie un contenu sur la sortie standard
- apache redirige ce contenu vers le navigateur

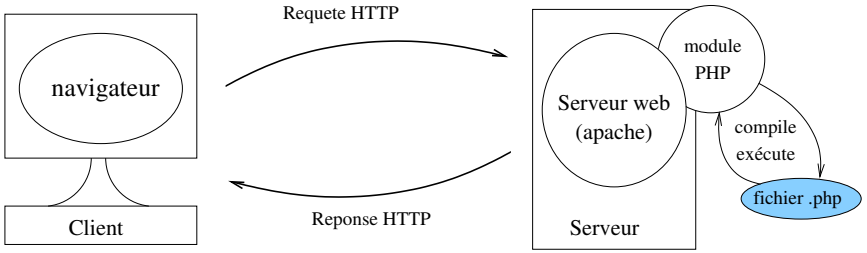
**Lent et lourd**

Il existe des variantes (FAST_CGI) pour améliorer les performances.



# Serveur : Modules et langages web

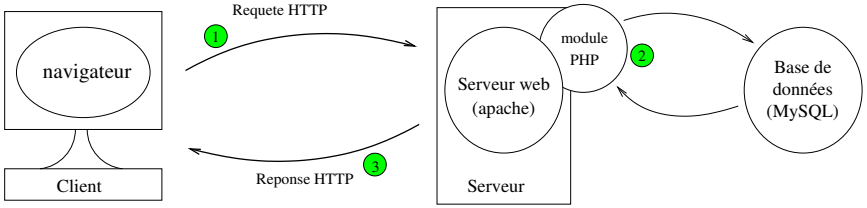
On remplace le CGI par une extension du serveur web (module).




- permet d'utiliser un langage interprété
- cette extension (module) est spécifique au serveur web.
  $\Longrightarrow$ seuls certains langages sont possibles :
  PHP, C#, Perl, Ruby, Python...



# Applications n-tier




3 parties : client — serveur — SGDB

Les données persistantes sont stockées dans MySQL.



# Rappels HTML

# (X)HTML, CSS, JS, etc.

## Exemple type

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Horloge</title>
</head>

<body>
<p>
Nous sommes le lundi 12 janvier 2154.<br />
Il est <strong>10 h 15</strong>.
</p>
</body>
</html>
```

Listing 1: typique.html



## Formulaires HTML

```
<form name="ex" action="suite.php" method="get">
    <fieldset>
        <legend>Groupe 1</legend>
        <label>Login :</label>
        <input name="login" type="text" />
        <label>Mot de passe :</label>
        <input name="mdp" type="password" />
        <label>Choix :</label>
        <select name="choix">
            <option selected="selected">1</option>
            <option value="2">deuxieme</option>
        </select>
        <button type="submit">OK</button>
    </fieldset>
</form>
```

Listing 2: form.html



## Recommandations

### Respecter les standards

- Choisir HTML4.01 (avec balises fermées) ou XHTML1 (véritable XML sur du HTML 4.01)
- Donner le DOCTYPE correspondant au navigateur
- Se référer aux spécifications : `http://www.w3.org/TR/html401/cover.html`

### Pour un HTML de qualité

- Séparer la présentation CSS du HTML : placer dans le `<link rel="stylesheet" type="text/css" href="..." />` `<style type="text/css">...</style>`
- Utiliser les attributs HTML *class* et *id* pour aider CSS et JS
- **Tester** sur plusieurs navigateurs dont au minimum IE7, IE8 et Firefox (+ Safari, Chrome, Opera ?)



# Le langage PHP

# Rendre dynamique le HTML

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Horloge</title>
</head>

<body>
<p>
Nous sommes le <?php
setlocale(LC_TIME, 'fr_FR');
echo strftime("%A %e %B %Y") ?>.<br />
Il est <?php echo strftime("%H h %M") ?>.
</p>
</body>
</html>
```



# PHP : qu'est-ce ?

## Les raisons du succès

- langage spécialisé : embarqué web
- simple (mais inspiré de C et de Perl)

## La maturité, enfin ?

- En évolution constante depuis sa création
  1994 : PHP 3 (première version publique)
  2000 : PHP 4 (ajout de la programmation objet)
  2004 : PHP 5 (programmation objet refaite !)
  2009 : PHP 5.3
- Débarrassé de quelques erreurs de jeunesses
  Certaines tentatives de simplification sont obsolètes en PHP5.
- Programmation objet possible : depuis PHP4 et surtout PHP5
- beaucoup de bibliothèques libres :
  e-mail, images, PDF, base de données...



# PHP : installation

## Logiciels nécessaires

- Serveur web (apache)
- Module PHP pour le serveur web (mod_php)
- Serveur de base de données MySQL
- Compléments optionnels : PHPMyAdmin...

## Sous Linux

Installer les paquets nécessaires. Par exemple, Debian/Ubuntu :

```
aptitude install apache2 libapache2-mod-php5 mysql-server
```

## Sous Windows

Choisir un installateur complet : **WampServer**, EasyPHP, XAMPP...



# Exemple

Coté serveur :

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.1 Strict//EN"
  "http://www.w3.org/TR/html41/DTD/html4.1-strict.dtd">
<html>
<head>
    <title>Exemple</title>
</head>
<body>
    <!-- un commentaire HTML -->
    <p>
        <?php echo "1789*19.6/100=" . (1789*19.6/100); ?>
    </p>
</body>
</html>
```

exemple.php, sur le serveur

Coté client (réponse à une requête HTTP) :

## PHP à grands traits

```php
<?php
if (2 > 1) {
    /* Le test ci-dessus devrait nous amener ici */
    echo "Tout va bien !";
} // fin du bloc du if
?>
```

- Le code PHP est toujours entre **<?php** et **?>**
  Éviter les "short tags" : <? ...?>
- Généralement sensible à la casse : `$a` ≠ `$A`
- instructions terminées par **;**
- blocs délimités par **{ ... }**
- commentaires avec **//** en fin de ligne ou **/* ... */**
- toujours se référer à la documentation officielle :
  `http://fr.php.net/manual/fr/`



## Première expérimentation

Le but est de réaliser une multiplication par un formulaire web.

1. Créer une page avec un formulaire permettant de saisir 2 nombres. Les noms des champs du formulaire seront "a" et "b".
2. Cette page est-elle en HTML ou en PHP ? Pourquoi ?
3. Ce formulaire doit être envoyé sur une page `res.php`.
   Y afficher le contenu de `$_GET['a']` et `$_GET['b']`.
4. Sur la page `res.php`, afficher l'opération avec son résultat, par exemple : $13x14 = 182$.



# Premiers pas en PHP

# Les données



## Variables

Préfixées par **$**
`$nombre = 174;`

### Déclaration

La déclaration n'est pas obligatoire, mais recommandée.

Une variable est créée à sa première utilisation. Par défaut, elle vaut toujours `NULL`.

Certaines variables spéciales sont créées par PHP avant l'exécution du script : `$_GET`, par exemple.

### Fonctions utiles

- `isset()` Teste l'existence
- `unset()` Supprime une variable
- `empty()` Teste si une variable est vide (inexistante, 0, "", etc.)

# Types de données

Langage *faiblement typé* (types non déclarés).
Conversion automatique des scalaires : `echo 5` $\Longleftrightarrow$ `echo "5"`

PHP connaît en interne différents types de données :

- scalaire entier : `-157`
- scalaire flottant : `15.7`
- scalaire chaîne : `"Ceci est un texte"`
- scalaire booléen : `TRUE, FALSE`
- tableau
- objet

## Fonctions utiles

- `var_dump($var)` : détaille le contenu de `$var`
- `gettype($var)`
- `is_integer($var)`, `is_string($var)`, etc.



# Les scalaires : les nombres

## Opérateurs numériques

**Identiques à ceux du C.**
Affectation : `=`
Calcul : `+ - * /`
Combiné : `+= -= *= /=`
Modulo : `%`
Comparaison : `== < <= > >= !=`
Incrémentation/décrémentation : `++ --`

Exemple :

```
$a = 1;
$a++;
$a += 4;
$a *= 2; // au final : 12
```



# Les scalaires : les chaînes de caractères

## Interpolation

- Sans interpolation

| | |
|---|---|
| `$t = 'Salut!';` | `Salut!` |
| `$t = 'Salut\n a \\toi';` | `Salut\n␣a␣\toi` |
| `$t = 'j\' arrive';` | `j'arrive` |

Tous les caractères sont conservés, sauf `\\` et `\'`.

- Avec interpolation

| | |
|---|---|
| `$t = "Salut\n a \\toi";` | `Salut`<br>`␣a␣\toi` |
| `$a = 'PHP';`<br>`$b = "Le site de $a";` | `Le␣site␣de␣PHP` |
| `$j = 40;`<br>`$t = "Les $j voleurs";` | `Les␣40␣voleurs` |



# Les scalaires : les chaînes de caractères

## Affichage

`echo` / `print` / `printf()`
Exemple : `echo '<p>La variable "$a"</p>';`

## Opérateurs

Concaténation : `.`
Comparaison : `== < <= > >= !=`

**Attention :**
**les opérateurs peuvent induire des conversions implicites.**
Les comparaisons suivantes valent-elles TRUE ou FALSE ?

- "précédent" $<$ "suivant"
- 8 $<$ "13"
- "21" $<$ "9"
- 9 $<$ "dix-huit"

Si on n'est pas sûr d'avoir une chaîne non numérique, utiliser `strcmp()` au lieu de `<`.

## En résumé : variables

- Une variable peut contenir des nombres...
  `$a = 15; $prix = 2.50;`
- ou du texte...
  `$txt = 'La monnaie est le $';`
  Interpolation : `$msgPrix = "Vendu pour $prix euros";`
- ou un booléen...
  `$vrai = true;`
- ou un tableau, un objet... (à suivre)
- Comparaisons et opérations
  `if ($a < 2) { $pluriel = true; }`



## En résumé : tests et boucles

PHP utilise la syntaxe du C :

```
// boucle pour $i allant de 0 a 99
for ($i=0 ; $i<100 ; $i++) {
    ...
}
// en PHP, on utilise rarement "for"
```

```
// le plus simple des tests
if ($a > 10) {
    ...
} else {
    ...
}
```

```
// idem serie de "if/else if/else"
switch ($var) {
    case 'a': ...
        break;
    default: ...
        break;
}
```

```
// on boucle tant que $i!=0
while ($i != 0) {
    ...
}
```



## Exercices

1. Créer une page PHP affichant “Hello world !” en utilisant 2 variables PHP. Proposer des variantes.
2. Qu'affichent les instructions suivantes ?
   ```
   $a=1;
   $b="2+$a";
   $a=2;
   echo $b . $a;
   ```
3. Que se passe-t-il quand on incrémente une variable inexistante ? Quand on l'affiche ? Expérimenter.
   Moralité : mieux vaut donner une valeur initiale à ses variables.
4. Calculer et afficher les tables de multiplication dans un tableau HTML.



## Attention aux comparaisons !

Qu'affiche le code suivant ?

```
if (3 == "3") { echo "1\n"; }
if (0 == "") { echo "2\n"; }
if ("FALSE" == TRUE) { echo "3\n"; }
```

Il faut pouvoir comparer exactement, **sans conversion** :

`===` et `!==`

```
if (3 === "3") { echo "1\n"; }            // FALSE
if (0 === "") { echo "2\n"; }             // FALSE
if ("FALSE" === TRUE) { echo "3\n"; }     // FALSE
```

Remarque :

```
if ($a) { ... };
if ($a == TRUE) { ... };  // idem ci-dessus
if ($a === TRUE) { ... }; // PLUS RESTRICTIF sur $a !
```

2 notations équivalentes :

```
if (empty($a)) { ... };
if (isset($a) && $a) { ... };
```

## Les tableaux

- plus précisément : **tableaux associatifs ordonnés**
- paires **clé / valeur** (dictionnaire) :

```
$tab = array( 0=>1, 1=>1, 2=>2, 3=>3, 4=>5 );
```

```
$tab = array( 0=>1, 1=>1, 2=>2, 3=>3, 4=>5,
              "nom" => "Fibonacci" );
```

- **accès aux valeurs** :

```
$tab["nombre"] = "d'or";
echo $tab["nombre"];
```

- cas des **tableaux numériques** : simplification
  Création abrégée (les clés sont 0, 1, 2...) :

```
$tab = array( 1, 1, 2, 3, 5 );
```

  **Ajout** en "fin de tableau"

```
$tab[] = "derniere valeure"; // ici $tab[5]
```

- multitude de fonctions pour les tableaux :
  `http://fr.php.net/manual/fr/function.array.php`



## Parcours d'un tableau

Pour un tableau à indice numérique, méthode classique :

```
for ($i=0 ; $i<count($tab) ; $i++) {
    echo "$tab[$i]\n";
}
```

Pour tout tableau :

```
foreach ($tableau as $valeur) {
    echo "$valeur\n";
}
foreach ($tableau as $cle => $valeur) {
    echo "$cle -> $valeur\n";
}
```

***foreach* est la boucle la plus fréquente en PHP !**



## Exercices

1. Peut-on interpoler un tableau dans une chaîne, c'est-à-dire écrire `echo "$tab[1]"` ou `echo "$tab['clef']"` ?
2. Que donne `echo $tab` et `echo "$tab"` ? Afficher en HTML tout le contenu d'un tableau PHP.
3. Expérimenter le code suivant. Qu'en déduire ?

```
$tab = array( 1, 2, 3 );
foreach ($tab as $ele) { $ele++; }
```

4. Placer des images dans un répertoire `"pics"` du serveur PHP, puis utiliser la fonction `glob('pics/*')` pour les afficher toutes dans une page HTML.
5. Écrire un tableau PHP de liste de titres de disques. L'afficher sous forme de liste HTML.
6. Transformer le tableau ci-dessus en liste de titres avec leur auteur/compositeur. L'afficher sous forme de <TABLE>.
7. Insérer un élément en queue de liste. Au début de la liste.
8. Supprimer le premier élément de la liste. Le dernier.



## Débogage basique

- `print_r($variable)` : afficher le contenu de $variable
- `var_dump($variable)` : idem, plus détaillé
- `die("message")` : arrête l'exécution du script en affichant ce message
- `error_log("message")` : insère un message dans le log du serveur http

**Activer les avertissements complets**

`error_reporting(E_ALL|E_NOTICE);`
signale les erreurs et les actions suspectes (utilisation variables non initialisées...)

**Un conseil :** placer cette ligne en début de fichier pendant la phase de développement.
Comportement par défaut sur le serveur en modifiant `php.ini`

# Formulaires HTML et données HTTP



## Éléments des formulaires HTML

- Ligne de saisie de texte
```
<input type="text" name="x" value="" />
```
- Zone de saisie multi-lignes
```
<textarea rows="5" cols="80" name="x"></textarea>
```
- Liste déroulante
```
<select name="x">
 <option value="id1">A</option><option>B</option>
</select>
```
- Liste déroulée
```
<select name="x" size="5" multiple="multiple">...
```
- Boutton poussoir envoyant le formulaire
```
<input type="submit" name="x" value="Envoyer" />
```
ou
```
<button type="submit" name="x" >Envoyer</button>
```

`http://www.w3.org/TR/html401/interact/forms.html`



## Formulaires HTML

```
<form action="http://localhost/formulaire.php"
  method="get">
<p>
    <input type="text" name="login" value="" />
    <button type="submit">OK</button>
</p>
</form>
```

Poster ce formulaire (rempli) se traduira par un accès à l'URL :
`http://localhost/formulaire.php?login=Albert`

Comment accéder à ces informations en PHP ?

`$_GET` : tableau associatif des paramètres GET
Pour ce formulaire : `$_GET['login']`



## Formulaires HTML et variables PHP superglobales

Variables superglobales (toujours accessibles)
- créées en interne par PHP (et non par le code utilisateur)
- débutant par `$_`

3 superglobales utiles pour les formulaires :
- `$_GET` : tableau des paramètres GET (reçus par l'URL)
- `$_POST` : tableau des paramètres POST (reçus par le champ POST de l'en-tête HTTP)
- `$_REQUEST` : fusion de ces 2 tableaux

Donc `$_REQUEST` convient que le formulaire utilise *method='GET'* ou *method='POST'*.

### Attention au vieux PHP

Avant PHP 4.3, la configuration *register_globals* était activée.
$\Longrightarrow$ PHP crée des variables au nom des paramètres GET et POST : `$_GET['a']` devient `$a`.
$\Longrightarrow$ Code sale + gros problèmes de sécurité !

## Exercices

1. Écrire une page d'authentification avec mot de passe caché et fixé dans le code.
2. Écrire un formulaire de saisie de disque. L'incorporer au projet précédent pour pouvoir ajouter un élément à la liste.
3. Valider le formulaire ci-dessus avant d'ajouter le disque, en demandant de compléter toute case vide.
4. Ajouter un choix de genre musical unique.
5. Comment faire si le disque appartient à plusieurs genres ? Quelle syntaxe HTML et PHP adopter ?
6. Utiliser la superglobale `$_SERVER` pour contrôler si le client est dans une liste d'IP valides.



# Fonctions



## Fonctions

### Déclaration

```
function add( $param1, $param2 ) {
    // ...
    return $param1 + $param2;
}
```

### Utilisation (appel)

```
echo add(88,-14) . add(3,4);
```

### En détails

- Renvoi (facultatif) d'une unique valeur (scalaire, tableau, etc.)
- Paramètres
  - en nombre fixe
  - sauf en cas de **valeur par défaut** avec la déclaration `function add($p1, $p2=1) { ...`
  - paramètre **modifiable** si déclaration sous la forme `&$param`



## Portée des variables

Les variables ont une zone d'action, dite **portée**.

```
$glob = "glob";
if (true) { $truc = "truc"; }
function test() {
    global $glob;
    $loc++;
    echo "$glob $truc $loc\n";
}
test();
echo "$glob $truc $loc\n";
test();
```

Affichera ?

- Dans une fonction, 3 types de variables utilisateur :

## Fonctions utiles pour du texte

Fonctions très nombreuses :
`http://fr.php.net/manual/fr/ref.strings.php`

Une sélection

- `htmlspecialchars` / `htmlentities` : passe du texte brut au HTML en remplaçant certains caractères par leur entité HTML
- `implode(" ",$tab)` : fusionne un tableau en une chaîne
- `explode(" ",$tab)` : découpe une chaîne en un tableau
- `trim($texte)` : supprime les espaces de début et fin
- `strtolower` / `strtoupper` : change la casse
- `strlen` : donne la taille d'une chaîne
- `strpos` : trouve la position d'un caractère dans une chaîne
- `substr` : retourne un segment de chaîne



## Inclusion de fichiers

Quand le code grossit, il faut le répartir dans plusieurs fichiers :
`include("fichier.php")` : le contenu de ce fichier est inséré ici.

Variantes de `include` :

- `include_once` : le fichier est inclus que s'il ne l'a pas encore été (utile pour les bibliothèques de code)
- `require` : en cas d'absence du fichier, arrêter tout sur une erreur
- `require_once` : fusion des 2 précédents, le plus utilisé et le plus pratique

**A retenir**
**`require_once`** pour charger les fichiers contenant des déclarations de fonctions.



## En résumé

- Il faut utiliser des fonctions :
  - code plus concis et plus lisible
  - moins de copier-coller $\Longrightarrow$ code maintenable
  - moins de risques de conflit de variables
- Les variables des fonctions n'existent que dans la fonction (variables **locales**).
- Pour accéder aux variables extérieures, utiliser `global`.
- Placer les déclarations de fonctions dans des fichiers séparés.
- Charger les fichiers de déclaration avec `require_once`.
  Syntaxe fréquente :
  `require_once dirname(__FILE__) . '/lib.php';`
- Chercher les fonctions dans la documentation officielle.



## Exercices

1. Écrire une fonction renvoyant la somme des éléments du tableau reçu en argument. Et si on ne reçoit pas de paramètre ? ou pas un tableau ?
2. Avec la fonction PHP `date()`, écrire une fonction renvoyant la date courante en anglais. Passer au français. L'adapter pour qu'on puisse optionnellement décaler la date de X jours.
3. Reprendre le code qui affiche un tableau HTML de disques et en faire une fonction à un paramètre de type tableau PHP.
4. Écrire une fonction qui valide une adresse IP :
   `check_ip('88.77.66.1','127.0.0.1 192.168.');`
5. Écrire une fonction qui valide une adresse IP par rapport à une plage CIDR :
   `ip_cidr('88.77.66.1','192.168.1.1/24');`

# Fonctions diverses
# e-mail, date, texte, PDF



## e-mails

### Exemple simple

```
$message = "Bonjour\nAu revoir";
mail(' destinataire@exemple . fr ', 'Mon Sujet', $message);
```

### Exemple complexe

```
$message = "Bonjour\nAu revoir";
$message = wordwrap($message, 70); // 70 car. max par ligne
$headers = 'From: anonyme@exemple.fr' . "\r\n" .
    'Cc: copie@exemple. fr ' . "\r\n" ;
mail(' destinataire@exemple . fr ', 'Mon Sujet', $message, $headers);
```

Pour des besoins complexes, mieux vaut utiliser une biblio dédiée, comme **PHPmailer** `http://phpmailer.codeworxtech.com/`.



## Gestion des dates

3 fonctions essentielles : (voir le manuel pour les détails)

- `date($format,$dateheure)` : formate une date/heure (par défaut, maintenant)
- `strftime($format,$dateheure)` : idem, localisé. Précéder cet appel d'un `setlocale(LC_TIME, "fr_FR");`
- `mktime(...)` : construit une dateheure (*timestamp*)

### Exercices

1. Convertir une date saisie par l'utilisateur au format français en date ISO. Accepter "23/10/2012" et "15/01/2001".



## Utiliser un objet en PHP

Un objet est une variable qui regroupe des données et des fonctions.

```
$objet = new MaClasseObjet();
$objet->attribut = "OK";
echo $objet->attribut;
$objet->methode(10);
```

### Exercices

1. Utiliser le système de "template" Savant3 pour piloter une page d'authentification.

## Constantes

S'utilise comme une variable de valeur scalaire fixe, non modifiable.

```
define('DEBUG',TRUE);

define('MA_CONSTANTE', 1.61828);
print(1+1/MA_CONSTANTE);
if (!defined('MA_CONSTANTE')) {
    define('MA_CONSTANTE', 2.712);
}
define('CARRE', MA_CONSTANTE*MA_CONSTANTE);

define('MA_CONSTANTE', 3.1415); // ERREUR
define('TABLEAU', array()); // ERREUR
```



## En-têtes HTTP

Un en-tête par défaut est envoyé, on peut le modifier avec `header(...)` uniquement **avant d'avoir envoyé des données**.

Quelques scénarios :

- La page n'est pas en HTML :
  ```
  header('Content-Type: text; charset=utf-8');
  header('Content-Type: application/pdf');
  ```
- La page ne doit pas être placée dans le cache navigateur
  ```
  header("Cache-Control: no-cache, must-revalidate");
  ```
- Redirection vers une autre page (l'URL de destination doit être complète et non locale)
  ```
  header("Location: http://serveur.fr/nouveau.html");
  exit();
  ```



## PDF

De nombreuses bibliothèques existent :

- La plus simple : **FPDF** (`http://www.fpdf.org/`)
  Possibilités limitées !
- Pour aller plus loin : **TCPDF** (`tcpdf.sourceforge.net`)
  Surcouche à FPDF avec gestion de l'UTF-8, modification de PDF existant (grâce au complément FPDI), etc.
- **Zend_Pdf** : récent, encore en évolution.

Pour être exhaustif, il existe 2 extensions référencées dans la doc officielle :

**libPDF** très limité dans sa version gratuite.

**Haru** libre, mais encore expérimental



## Sessions

Comment garder des données utilisateur entre 2 exécutions d'un script ?

```
<?php
$compteur++; // vaudra toujours 1
echo "Nombre de visites connues : $compteur";
```

normal.php

```
<?php
session_start();
$_SESSION['compteur']++;
echo "Nombre de visites connues : $_SESSION[compteur]";
```

session.php

Une session permet de rendre des données PHP persistantes entre les requêtes d'un même utilisateur.

### Exercices

Le but est de créer un fichier de fonctions PHP qui permettent de :

1. Construire 2 pages web, "page1.php" et "page2.php". Chacune doit afficher quelle est la dernière page visitée ("aucune", "page1" ou "page2").
2. Sur une page d'authentification qui valider un login et un mot de passe, utiliser une session pour garder le statut authentifié d'un utilisateur en dehors de la page de login. Afficher "Bonjour, " en haut de la page listant les disques.
3. Rediriger vers une page d'avertissement si on demande une page sans être authentifié.
4. Ajouter sur une page un lien HTML permettant de se déconnecter.



# PHP & MySQL



## Généralités

### Rappel : pourquoi une base de données dans PHP ?

- Données persistantes de l'application
- Facilités de traitement (à comparer avec un stockage fichier fait soi-même)

Ne pas confondre avec les sessions (individuelles).

### Déroulement

1. Le script PHP ouvre une connexion à un serveur MySQL
2. Plusieurs requêtes possibles pour une même connexion
3. Les résultats MySQL sont reçus ligne par ligne dans des tableaux PHP
4. Les connexions MySQL se ferment à la fin du script PHP



## PHP4 & MySQL (obsolète)

### Initialisation de la connexion

```
$dblink = mysql_connect('serveur','login','motdepasse');
if (! $dblink)
    { die( "ERREUR de connexion : " . mysql_error() ); }
mysql_query("SET NAMES 'latin1'"); // ou 'utf8'
if (! mysql_selectdb('BaseDeDonnees'))
    { die( "ERREUR d'acces a la base : " . mysql_error() ); }
```

### Lire des données

```
$resultat = mysql_query("SELECT * FROM Disques");
if (! $resultat)
    die('Erreur dans la requete : ' . mysql_error());
echo "Il y a ".mysql_num_rows($resultat)." reponses.\n";
while ($ligne = mysql_fetch_assoc($resultat)) {
    echo $ligne['titre'] ;
    print_r($ligne); // utiliser l'enregistrement
}
```

# PHP5 & MySQL

Il est recommandé d'utiliser **mysqli** à la place de *mysql*.
On peut remplacer *mysql_* par *mysqli_*, ou utiliser la syntaxe objet ci-dessous.

## Initialisation de la connexion

```
$db = new mysqli('serveur','login','motdepasse','BaseDeDonnees');
if (! $db)
    { die( "ERREUR de connexion : " . mysqli_connect_error() ); }
$db->set_charset("utf8"); // ou 'latin1'
```

## Lire des données

```
$resultat = $db->query("SELECT * FROM Disques");
if (! $resultat)
    die('Erreur dans la requete : ' . $db->error);
echo "Il y a " . $resultat->num_rows . " reponses.\n";
while ( $ligne = $resultat->fetch_assoc() ) {
    echo $ligne['titre']; // utiliser l'enregistrement
}
```



# En résumé, MySQL en PHP5/mysqli

Procédure usuelle avec **mysqli** :

1. Initialisation de la connexion (souvent au début de la page)
   ```
   $db = new mysqli('serveur','login','mdp','Base');
   $db->set_charset("utf8"); // ou 'latin1'
   ```
2. Envoi d'une requête au serveur MySQL
   ```
   $resultat = $db->query("SELECT * FROM Disques");
   ```
3. Si la requête est un SELECT, lecture ligne par ligne du résultat reçu
   ```
   while ($ligne = $resultat->fetch_assoc() ) {
   ```
4. La connexion est automatiquement fermée en fin de page.



# En résumé, MySQL en PHP5.1/PDO

Procédure usuelle avec **PDO** :

1. Initialisation de la connexion (souvent au début de la page)
   ```
   $db = new PDO('mysql:host=serveur;dbname=Base','login','mdp');
   $db->query('SET NAMES UTF8');
   ```
2. Envoi d'une requête au serveur MySQL
   ```
   $resultat = $db->query("SELECT * FROM Disques");
   ```
3. Pour un SELECT, lecture ligne par ligne du résultat reçu
   ```
   while ($ligne = $resultat->fetch(PDO::FETCH_ASSOC)) {
   ```
   Autre syntaxe :
   ```
   foreach ($resultat as $ligne) {
   ```
   On peut aussi récupérer un tableau complet des lignes
   ```
   $lignes = $resultat->fetchAll(PDO::FETCH_ASSOC);
   ```
4. La connexion est automatiquement fermée en fin de page.



# Compléments : Variantes du *fetch*

## Avec mysqli

fetch_assoc tableau associatif : `$ligne['date']`
fetch_row tableau numérique : `$ligne[2]`
fetch_object objet : `$ligne->date`

## Avec PDO

Pour les fonction `fetch*`, paramètre optionnel parmi :
PDO::FETCH_ASSOC
PDO::FETCH_NUM
PDO::FETCH_OBJ

## Mise en application

1. Afficher un tableau HTML rempli avec le contenu de la table *Disques*. Transformer ce code en une fonction PHP qui reçoit un nom de table à afficher.
2. Créer une table MySQL d'utilisateurs (login + mot de passe). Écrire une page PHP d'authentification l'utilisant. Conserver le statut "authentifié" d'un utilisateur sur d'autres pages.
3. Faire un formulaire pour ajouter un artiste à la table. Le modifier pour permettre aussi la modification d'un artiste.
4. Faire un formulaire de saisie de disque pour ajout dans la base. Le genre et l'artiste seront choisis dans une liste déroulante.
5. Modifier le point précédent pour autoriser un choix de multiples genres. Comment en faire autant pour les artistes ?
6. Écrire un formulaire de recherche dans les CD. D'abord avec seulement titre et date, puis label, puis genre et artiste.
7. Paginer les résultats des recherches.
8. Ajouter dans une table un log sur les actions effectuées.



## Compléments : LAST_INSERT_ID

Après cette requête
`INSERT INTO Artiste SET nom='Rolling stones'`
comment connaître l'identifiant AUTO_INCREMENT créé ?

- en SQL
  `SELECT LAST_INSERT_ID();`
- en PHP 4 (mysql)
  `mysql_insert_id()`
- en PHP 5 (mysqli)
  `mysqli_insert_id()` ou `$db->insert_id`
- en PHP 5 (PDO)
  `$db->lastInsertId()`



## Un peu de sécurité

Il faut distinguer nombres et chaînes :

- Un nombre peut être utilisé tel quel
- Une chaîne doit être **protégée** et placée entre '...'

### Exemple en mysqli :

```
$db->query("SELECT * FROM matable "
    ."WHERE id_table=$number "
    ." OR titre='" . $db->real_escape_string($texte)."'");
```

(mysql utilise `mysql_real_escape_string()`)

### En PDO :

```
$db->query("SELECT * FROM matable "
    ."WHERE id_table=$number "
    ." OR titre=" . $db->quote($texte));
```



## La protection automatique de PHP

Deux modes de fonctionnement très différents :

- **Magic quotes** OFF
  Les données HTTP ne sont pas modifiées :
  `echo $_GET['text'];` $\longrightarrow$ Un'Deux"Trois\
- **Magic quotes** ON
  On applique `add_slashes()` aux données HTTP :
  `echo $_GET['text'];` $\longrightarrow$ Un\'Deux\"Trois\\

On peut connaître la valeur avec `get_magic_quotes_gpc()`.
Ne peut se changer à la volée (cf fichier `php.ini`).

$\Longrightarrow$ Un code portable doit au minimum vérifier cette valeur.

**Protection obsolète !** Voir la doc à ce sujet.

## Mise en application (suite)

1. Tester l'extension PHP xdebug.
2. Écrire une application de forum simplifié
   - une page de saisie du message, avec le corps du texte et soit le titre (pour une nouvelle discussion), soit le sujet (pour un ajout à une discussion existante) ;
   - une page d'affichage de tous les messages

   Vérifier sa sécurité (injection SQL, XSS).



## Création d'un site PHP

Un développement est très rarement *from scratch*, on utilise des composants éprouvés :

CMS *Drupal, Joomla, Wordpress...*
Application complète avec d'innombrables extensions. Facilite la publication d'information par des non-développeurs.

framework *Zend Framework, Symfony, Yii...*
Permet un développement rapide et structuré.

bibliothèque *PEAR, eZcomponents...*
Par exemple, *eZFeed* pour lire ou créer des flux RSS.

**Exceptions**

Sites extrêmement simples,
ou très spécifiques (par exemple, webservices).



## Informations utiles

Pour garder le contact :
`francois.gannaz@silecs.info`

Les documents utilisés sont disponibles en ligne :
`http://silecs.info/formations/PHP-MySQL/`

- Transparents
- Corrections des exercices



## Licence