PHP (ET MYSQL)

Introduction

- HTML : pages destinées à être publiées sur Internet
  - Texte à afficher + instructions de mise en page
  - Pas d'instructions de calcul ou de traitements conditionnels
- Des sites de plus en plus riches en informations
  - Nécessité croissante d'améliorer le contenu des sites
  - Mises à jour manuelles trop complexes
    - Exemple : modifier l'entête sur plusieurs pages !
  - Besoin de réponses spécifiques liées à un BD par exemple
- Passage de sites statiques à des sites dynamiques

Web dynamique – coté client

- Traité par le navigateur :
  - Résultats variables en fonction du navigateur
    - Nécessité de tests importants
  - Indépendant du serveur
    - Pas de rechargement de la page, tout est fait en local

- Confiance :
  - Sources du programme disponibles
  - Données envoyées au serveur pas fiables
  - Base de données stockée chez le client ?

Web dynamique – coté serveur

- L'interprétation est réalisée par le serveur :
  - Indépendant de la machine et du navigateur
  - "Compatible" avec tous les navigateurs
    - Les échanges ne concernent que du HTML (ou Json ou autre)
  - Les sources sont sur le serveur donc invisibles

- Besoin d'échanges entre le navigateur et le serveur
  - Rechargement de la page à chaque modification
  - Ou Ajax…

Web dynamique – client ou serveur

- Il faut les deux !

- Script côté client (Javascript) :
  - Calculs et traitement simples
  - Mises à jour de la page web sans rechargement (ajax)

- Script côté serveur (Php ou autre) :
  - Calculs, traitements plus conséquents
  - Requêtes vers une base de données
  - Opérations sécurisées

Le langage Php - histoire

- PHP : Hypertext PreProcessor

- La première version de PHP (Personal Home Pages) sort en 1995
- La version 5.0 actuelle est sortie en 2004 :
  - la version 4.0 n'est plus maintenue depuis août 2008.

- Environ 75% des sites web dynamiques utilisent php.

Le langage Php

- Langage pour la création d'applications Web
- Exécution coté serveur
  - Indépendant de la plate-forme utilisée
  - Facile à combiner avec un serveur Web et une BD
    - Logiciels tout en un (WampServer, EasyPhp, …).
- Gratuit, code source disponible
- Ecriture de scripts simples
- Inclusion directe de Php dans du HTML

Le langage Php

- Programme s'exécutant côté serveur Web
  - Extension .php pour les pages PHP
  - Fichiers pouvant aussi contenir du HTML

- Les fichiers .php sont stockés sur le serveur
  - Désignés par une URL : http://www.lip6.fr/page.php
  - Le chargement de l'URL provoque l'exécution côté serveur
  - Le script Php va généralement créer du HTML

Exemple

- On veut créer la page web ci-dessous
  - Pour un nombre quelconque de valeurs (venant d'un formulaire par exemple).

```
<html>
<body>
1 : 1 <br />
2 : 4 <br />
3 : 9 <br />
</body>
</html>
```

Version Javascript

- Insertion de javascript avec la balise <script>
  - Boucle for classique
  - Affichage du contenu via document.write
  - Concaténation de chaînes avec le signe +
- Exécuté par le navigateur

```
<html>
<body>
<script type="text/javascript">
for(var i=1;i<=3;i++)
  document.write(i+" : "+i*i+"<br />");
</script>
</body>
</html>
```

Version Php

- Insertion de Php avec la balise <?php
  - Boucle for classique
  - Affichage du contenu via echo
  - Concaténation de chaînes avec le signe
- Exécuté par le serveur puis envoyé au navigateur

```
<html>
<body>
<?php
  for ($i=1 ; $i<=3 ; $i++)
    echo $i." : ".($i*$i)."<br />";
?>
</body>
</html>
```

Différence entre Javascript et Php

- Javascript :
  - Le navigateur demande le fichier html
  - Le fichier html (contenant du javascript) est envoyé
  - Le navigateur exécute le javascript
  - L'affichage est fait de manière dynamique

- Php :
  - Le navigateur demande le fichier php
  - Le serveur exécute le code php et envoie le résultat
  - La page envoyée au navigateur est affichée

Code Php – autre version

- Seules les parties purement dynamiques sont faites en Php
  - Tout le reste est du html de base
  - Pas forcément très lisible

```
<html>
<body>
<?php
for ($i=1 ; $i<=3 ; $i++) {
?>
<?php echo $i ?> : <?php echo $i*$i; ?><br />
<?php
}
?>
</body>
</html>
```

Code Php – autre version

- Tout le fichier html est créé en Php :
  - Syntaxe assez lourde

```
<?php
echo "<html>\n";
echo "<body>\n";
for ($i=1 ; $i<=3 ; $i++ ) {
 echo $i." : ".($i*$i)."<br />\n";
}
echo "</body>\n";
echo "</html>\n";
?>
```

PHP - SYNTAXE

VARIABLES

Variables

- Une variable est toujours précédée de $
  - $i est une variable qui s'appelle i
- Types :
  - Entiers, réels, chaînes de caractères, objets
- Pas de déclaration explicite, l'affectation d'une valeur suffit
  - Affectation par valeur : $i=0
  - Affectation par (référence) variable : $j = &$i
    - la modification de $i modifie $j (comme les pointeurs en C)
- Changement de type automatique :
  - $compteur="premier";
- Valeur par défaut avant l'affectation
  - Attention si on essaye de lire son contenu !

Variables

- Variable locale
  - Visible uniquement à l'intérieur d'un contexte d'utilisation
- Variable globale
  - Visible dans tout le script
  - Utilisation de l'instruction "global" pour accéder à une variable globale dans des contextes locaux

```
$var = 1;
function test(){
  global $var;
  echo $var;
}
```

Chaînes de caractères

- Création/affectation entre guillemets
  - $test="a";
- Concaténation avec le signe .
  - $test="bonjour" . "au revoir";

- Substitutions possible de variables à l'intérieur d'une chaîne :
  - "x $test x" est similaire à "x ".$test." x"
  - pas de substitution avec des apostrophes
    - 'x $test x'; // "x $test x"

- Encodages des caractères spéciaux obligatoire :
  - \$ \\ \n \t \"…

Création de variables dynamiques

- Possibilité de créer une variable dynamique :
  - Création à partir du contenu d'une autre variable.

```
$var="test";
$$var="test2"; // similaire à $test="test2";

echo "$var $test ${$var}"; // affiche "test test2 test2"

// similaire avec des tableaux
$nom_variable = array("val0", "val1");
${$nom_variable[0]} = "x";
```

Opérations sur les chaînes

- Longueur d'une chaîne strlen
  - strlen($str)
- Comparaison ==
  - $x == $y
- Concaténation .
  - $x.$y
- Nombreuses fonctions de manipulation disponibles

Fonctions sur les chaînes

- Afficher une chaîne de caractères :
  - echo $chaine;
- Retourner un morceau d'un chaîne : substr($chaine, $debut, $longueur)
  - $debut peut être négatif : en partant de la fin
  - $longueur est facultatif : toute la fin de la chaîne est retournée
- Couper une chaîne en morceaux avec un délimiteur :
  - $tableau = explode($delimiteur, $chaine);
- …

INSTRUCTIONS

Instructions conditionnelles

- if then else
  - définition spéciale pour le else if (pas obligatoire)

```
if (cond) {
 ...
} elseif (cond) {
 ...
} else {
  ...
}

// version courte :
(condition)?instructionSiVrai:instructionSiFaux;
```

Instructions conditionnelles

- switch (…) { case … : … ; }

```
switch (expression) {
   case "1" : ... ; break;
   case "2" : ... ; break;
   case "3" : ... ; break;
   default : ...
}
```

Boucles

- Boucles for :
  - for (initialisation ; condition ; increment) { ... }
- Boucles while :
  - while (condition) { ... }
  - do { ... } while (condition);

Instructions conditionnelles

- Saut inconditionnel
  - continue : termine l'itération courante de la boucle
- Arrêt inconditionnel
  - break : termine la boucle complètement
- Arrêt d'exécution du script
  - exit : termine le script

```
for ($i=1; $i<=10; $i++) {
  if ($i == 5)
    continue;
  echo $i."<br />";
}

// affiche les valeurs sauf 5
```

```
for ($i=1; $i<=10; $i++) {
  if ($i == 5)
    break;
  echo $i."<br />";
}

// affiche jusqu'à 5 (exclu)
```

TABLEAUX

Les tableaux

- Création à l'aide de la fonction array()
  - $tableau = array(1, "deux", 3);
- Tableaux à une dimension
  - Les éléments peuvent être de différents types
  - L'index d'un tableau commence à 0
- Fonction :
  - count() pour avoir le nombre d'éléments d'un tableau

Tableaux simples

- Association numéro - valeur

```
// Création du tableau
$tableau = array(valeur0, valeur1,..., valeurN);

// Affectation d'un élément
$tableau[indice] = valeur;

// Récupération d'un élément
$variable = $tableau[indice];

// Parcours version 1
for ($i=0; $i<count($tableau) ; $i++)
  echo $i." ".$tableau[$i]."<br />";

// Parcours version 2
foreach($tableau as $valeur)
  echo $valeur."<br />";
```

Tableaux associatifs

- Associations clé d'index - valeur

```
// Création du tableau
$tableau = array(cle0 => val0, cle1 => val1,..., cleN => valN);

// Affectation d'un élément
$tableau["cle"] = valeur;

// Récupération d'un élément
$variable = $tableau["cle"];

// Parcours version 1
foreach($tableau as $cle => $valeur)
  echo $cle." ".$valeur."<br />;

// Parcours version 2
reset($tableau);
while (list($cle, $valeur) = each($tableau))
 echo $cle." ".$valeur."<br />;
```

Tableaux multidimensionnels

- Pas de méthode de création native :
  - On imbrique des tableaux.

```
// Création d'un tableau à deux dimensions
$tab1 = array(Val0, Val1,..., ValN);
$tab2 = array(Val0, Val1,..., ValN);
$tableau = array($tab1, $tab2);

// Affectation d'un élément
$tableau[indice][indice] = $variable;

// Récupération d'un élément
$variable = $tableau[indice][indice];

// Parcours
for ($i=0 ; $i<count($tableau) ; $i++)
  for ($j=0 ; $j<count($tableau[$i]) ; $j++)
    echo $i." ".$j." ".$tableau[$i][$j]."<br />";
```

Fonctions sur les tableaux

- Taille d'un tableau :
  - count()
- Compter le nombre d'occurrences des valeurs d'un tableau :
  - $tab2 = array_count_values($tab);
- Tri :
  - sort($tab); // selon les valeurs
  - ksort($tab); // selon les clefs
- Mappage par une fonction (applique la fonction à tous les éléments) :
  - $tab2 = array_map("fonction", $tab);
- Filtrage par une fonction (supprime les éléments ne vérifiant pas le critère) :
  - $tab2 = array_filter($tab, "fonction")
- Intervertir les clefs et valeurs :
  - $tab2 = array_flip($tab);

FONCTIONS

Fonctions

- Pas de typage
  - Arguments
  - Valeur de retour
- Récursivité possible

```php
<?php

// définition de la fonction factorielle
function factorielle($n) {
  if ($n<2)
    return 1;
  else
    return $n*factorielle($n-1);
}

// appel de la fonction et afficahage du résultat
echo factorielle(3);

?>
```

Fonctions

- Valeurs par défaut possibles pour les arguments
  - Tous les arguments peuvent avoir des valeurs par défaut
  - function racine($x, $racine=2) {}
  - Appel : racine($a, $b) ou racine($a)
- Valeurs par défaut uniquement pour les derniers arguments
  - racine($x=4,$degre) impossible

```
// si un seul argument alors $degre vaut 2 par défaut
function racine($x,$degre=2){
  return pow($x,1/$degre);
}

racine(27,3); // retourne la racine cubique de 27
racine(16); // retourne la racine carrée de 16
```

Fonctions

- Nombre d'arguments inconnu :
  - func_num_args() : nombre d'arguments
  - func_get_arg($i) : argument i.
    - Numérotés à partir de 0

```
// fonction calculant le produit de tous les arguments
function produit() {
  $prod=1;
  for ($i=0 ; $i < func_num_args() ; $i++)
    $prod *= func_get_arg($i);
  return $prod;
}

echo produit(3, 77, 10, 5, 81, 9);
```

Fonctions

- Passage de paramètres par valeur :
  - Utilise une copie des paramètres d'appel.

- Passage de paramètre par référence :
  - Utilise directement les paramètres d'appel (donc modifiable).
  - Passage de paramètre avec &

```
function double($val) {
   $val *= 2;
   return $val;
}

$x = 10;
$y = double($x); // y=20 x=10
$y = double(&$x); // y=20 x=20
```

Variables globales et statiques

- Static permet de conserver une variable dans une fonction

```
$v = 2;

function test () {
  global $v; // variable globale
  static $x=0; // variable statique
  $x+=$v;
  echo $x."<br />";
}

test(); // affiche 2
test(); // affiche 4
```

Appel dynamique

- Similaire aux variables dynamiques

```
function bonjour(){
  echo "bonjour<br />" ;
}

function bonsoir (){
  echo "bonsoir<br />" ;
}

// création d'un variable contenant le nom d'une fonction
$salut = (date("h") <= 17)?"bonjour":"bonsoir";

$salut();
```

Fonctions de fonctions

- Vérifie l'existence d'une fonction :
  - function_exists("fonction");
- Retourne la liste des fonctions définies :
  - $tableau = get_defined_functions();
- Enregistre une fonction à exécuter à la fin du script :
  - register_shutdown_function("fonction");
- …

Fonctions utiles

- Header :
  - Permet de modifier l'entête du protocole http
  - Utilisation principale : redirection vers une autre page

```
if (isConnected($user)) {
  header("Location: accueil.php");
} else {
  header("Location: login.php");
}
```

LES FORMULAIRES

Exploitation d'un formulaire

- Accès à la page du formulaire
- Remplissage/modification des champs
- Envoi du formulaire(submit)

Client

- Récupération du formulaire soumis
- Traitement du formulaire
  - Calculs
  - Transformations
  - Accès aux BD …
- Envoi de la réponse au client

Serveur

Récupération des paramètres

- Les paramètres sont récupérés dans un tableau :
  - $_POST["nom"] si la méthode d'envoi est POST
  - $_GET["nom"] si la méthode d'envoi est GET
  - <form method="POST" action="…">

- Exemple
  - Dans le formulaire <input name="nom" >
  - On récupère le contenu en php avec :
    - $_POST["nom"]
    - $_GET["nom"]

Pour continuer

- Le plus simple pour comprendre :
  - Utiliser la méthode GET pour passer les arguments
  - Regarder l'URL appelée
    - http://www.test.com/test.php?champ=valeur
    - Les différents paramètres envoyés et leur valeur sont visibles
    - Ici $_GET["champ"] vaut valeur
  - Puis utiliser POST une fois qu'on a compris

- Spécificités selon les types :
  - les cases à cocher sont envoyées sous forme de tableau

Un exemple

```
<form method="GET" action="test.php">
<input type="text" name="Champ1" value="Texte" /><br />
<textarea name="Champ2" cols="30" rows="5">Texte</textarea> <br />
<select name="Champ3">
  <option value="Option_1">Option_1</option>
  <option value="Option_2" selected="selected">Option_2</option>
</select><br />
<input type="checkbox" name="Champ4[]" value="Case_1" checked="checked"> Case 1<br>
<input type="checkbox" name="Champ4[]" value="Case_2" checked="checked"> Case 2<br>
<input type="radio" name="Champ5" value="Case_1"> radio 1<br>
<input type="radio" name="Champ5" value="Case_2"> radio 2<br>
<input type="radio" name="Champ5" value="Case_3"> radio 3<br>
<input type="submit" name="Soumission" value="Soumettre">
</form>


test.php?Champ1=Texte&Champ2=Texte&Champ3=Option_2&Champ4[]=Case_1&Champ4[]=Case_2&Champ
    5=Case_2&Soumission=Soumettre
```

Un exemple

- Récupération avec $_GET
  - Vérification de l'existence d'une variable :
    - isset(var)
- Attention au contenu des choses reçues
  - Nombreuses failles si on ne fait pas attention !

```
// affichage de tous les champs reçus via le formulaire

$resultat = $_GET["Champ1"]."<br>";
$resultat .= $_GET["Champ2"]."<br>";
$resultat .= $_GET["Champ3"]."<br>";
for ($i = 0; $i < count($_GET["Champ4"]); $i++) {
  $resultat .= $_GET["Champ4"][$i]."<br>";
}
if (isset($_GET["Champ5"]))
  $resultat .= $_GET["Champ5"]."<br>";
echo $resultat;
```

FONCTIONNALITÉS SUPPLÉMENTAIRES

Date et heure

- Retourner la date courante dans une chaîne de caractères :
  - $chaine = date(format [, nombre]);
  - $chaine=date("Y/m/d"); retourne 2010/03/10
- Idem pour un tableau associatif :
  - $tableau = getdate([nombre]);
- Vérification de la validité d'une date :
  - checkdate(mois, jour, année);

Les cookies

- Écrire des cookies :
  - fonction setcookie("PremierCookie", "Salut", time()+3600*24*7) ;
    - timestamp en secondes (différent de Javascript)
    - à exécuter avant tout autre envoi vers le serveur (envoyé dans l'entête)
    - cookie non visible avant le prochain chargement d'une page
- Lecture :
  - ($_COOKIE["PremierCookie"]

```
setcookie ("PremierCookie", "Salut", time() +3600*24*7) ;

if (!isset($_COOKIE["PremierCookie"])) {
  echo "cookie non défini<br />";
} else {
  echo $_COOKIE["PremierCookie"]."<br />";
}
```

Les sessions

- Objectif : garder des données de page en page.
  - session_start()
    - Crée une nouvelle session
    - Ou ravive une session déjà existante
  - $_SESSION["nom"]=valeur permet de créer une variable de session
  - session_destroy() détruit la session en cours

```
session_start();
$_SESSION["ma_variable"]=12;
```

Sessions – exemple

```
<html>
<body>
<form method="post" action="test.php">
  <table border="0">
  <tr>
    <td>Nom :</td>
    <td><input type="text" name="Nom" size="20" value="x"></td></tr>
  <tr>
    <td>Prénom :</td>
    <td><input type="text" name="Prenom" size="20" value="y"></td></tr>
  <tr>
    <td>eMail :</td>
    <td><input type="text" name="Email" size="20" value="z"></td></tr>
  <tr>
    <td colspan="2"><input type="submit" name="soumettre" value="Envoyer"></td>
  </tr>
</table>
</form>
</body>
</html>
```

Sessions – exemple "test.php"

```php
<?php

session_start();

$_SESSION["nom"] = $_POST["Nom"];
$_SESSION["prenom"] = $_POST["Prenom"];
$_SESSION["email"] = $_POST["Email"];

?>
```

Sessions – exemple "session.php"

```
<html>
<body>

<?php
session_start();
echo "Identifiant :".session_id()."<br />";
echo "Nom de la session :".session_name()."<br />";
echo "Nom : ".$_SESSION["nom"]."<br>";
echo "Prénom : ".$_SESSION["prenom"]."<br>";
echo "Mail : ".$_SESSION["email"]."<br>";
session_destroy();
?>

</body>
</html>
```

Envoyer des emails

- La fonction mail :
  - mail($recipient, $subject, $message[, $headers, $params]);
  - Nécessite un serveur mail accessible sur le serveur

```php
<?php

$dest = "Moi <moi@moi.com>, Toi <toi@toi.com";
$subject = "Test";
$content = "Bonjour,\n";
$content .= "Ceci est un message de test\n";
$content .= "Moi";
$headers = "From: Moi <moi@moi.com>\n";
$headers .= "Content-Type: text/html; charset=iso-8859-1\n";
$headers .= "Cc: lui@lui.com\n";
mail($dest, $subject, $content, $headers);

?>
```

Créer autre chose que du HTML

- Php peut créer autre chose que du HTML
  - Images :
    - grand choix de formats, comme GIF , PNG , JPEG , WBMP…
  - Fichiers pdf (librairie tcpdf), csv, word, excel, …
  - Aucune limitation, il suffit de savoir comment modifier l'entête (header) puis afficher le contenu…

```php
<?php
header("Content-type: image/png");
$im = imagecreatefrompng("test.png");
$im = imagerotate($im, 90, 0);
imagepng($im);
?>
```

PHP ET MYSQL

Bases de données

- Présentation :
  - HTML, Javascript
  - Navigateur : IE, Firefox, Chrome, …
- Traitements / création de la page :
  - PHP, ASP, JSP, servlet, CGI, …
  - Serveur Web Apache, IIS, …
- Données :
  - Tables SQL
  - Serveur BD Access, Oracle, MySQL, PostGreSQL, …

Bases de données

- PHP permet de travailler nativement avec la plupart des SGBDR
  - Mysql, Oracle, Sybase, Microsoft SQL Server, PostgreSQL, …
  - Dans les autres cas on peut utiliser des drivers spécifiques

- Trois fonctions sont essentielles:
  - Connexion au serveur
  - Exécution de la requête SQL
  - Gestion des résultats

- On va voir le fonctionnement avec les "PHP Data Objects" :
  - Indépendant (partiellement) du SGBD utilisé.

MySQL et Php – connexion

- Connexion = création d'un objet PDO :
  - Adresse de la base et nom de la table
  - Nom d'utilisateur
  - Mot de passe
- Déconnexion : destruction de l'objet

```
try {
  // ouverture de la connexion
  $dbh = new PDO('mysql:host=127.0.0.1;port=3306;dbname=test', 'root', '');

  ...

  // fermeture de la connexion
  $dbh = null;
} catch (Exception $e) {
  die('Erreur : ' . $e->getMessage());
}
```

MySQL et Php – requêtes

- Avec :
  - la méthode query pour les recherches
  - Le méthode exec pour les modifications
- Libération des ressources :
  - closeCursor();

```
$dbh = new PDO('mysql:host=127.0.0.1;dbname=test', 'root', '');
$reponse = $dbh->query('SELECT * FROM `test`');
$nb_modifs = $dbh->exec('UPDATE `test` SET test_id=21');

...

$reponse->closeCursor();
$dbh = null;
```

MySQL et Php – traitement

- Nombre de réponses :
  - rowCount()
- Traitement des réponses :
  - fetch() : retourne les résultats un par un
  - fetchAll() : retourne un tableau avec tous les résultats

```
if ($reponse->rowCount() > 0) {
  while ($donnees = $reponse->fetch()) {
    echo $donnees['test_id'].' '.$donnees['test_val'].'<br />';
  }
} else {
  echo "aucun résultat<br />";
}
```

MySQL et Php – Un exemple

```
<html><body>
<ul>
<?php
try {
  $dbh = new PDO('mysql:host=127.0.0.1;dbname=test', 'root', '');
  $id=$_POST['id'];
  $reponse = $dbh->query('SELECT * FROM `test` WHERE test_id='.$id);
  if ($reponse->rowCount() > 0) {
    while ($donnees = $reponse->fetch()) {
      echo "<li>".$donnees['test_id'].' : '.$donnees['test_val'].'</li>';
    }
  } else {
    echo "<li>aucun résultat</li>";
  }
  $reponse->closeCursor();
  $dbh = null;
} catch (Exception $e) {
  die('Erreur : ' . $e->getMessage());
}
?>
</ul>
</body></html>
```

Sécurité

- Que se passe-t-il si :
  - $id = "12"
  - $id = "12 OR 1"
  - $id = "12; DROP TABLE `test`;"

```
$id=$_POST['id'];
$reponse = $dbh->query(
  'SELECT * FROM `test` WHERE test_id='.$id
);

SELECT * FROM `test` WHERE test_id=12
SELECT * FROM `test` WHERE test_id=12 OR 1
SELECT * FROM `test` WHERE test_id=12; DROP TABLE `test`;
```

Requêtes avancées

- Préparation de requêtes génériques :
  - prepare($query);
  - execute avec arguments
  - Plus sécurisé que la version précédente, mise en cache, plus portable… en clair à utiliser

```
$req = $dbh->prepare('
  SELECT test_id, test_val
  FROM `test`
  WHERE test_id = ? AND test_val= ?
');
$req->execute(array($x, $y));
```

Remarques finales

- Limiter au maximum les droits de l'utilisateur.
- Les données transmises par le client ne sont pas fiables :
  - SQL Injection
  - Javascript injection
    - Tenter de mettre javascript:alert("Hello!"); dans un input
- Toujours tester l'existence / la validité d'un fichier / code à inclure.
- Regarder régulièrement les logs.

MYSQL

MySQL – types de données

- INT
- FLOAT
- TEXT
- DATETIME / TIMESTAMP
  - Date et heure / Horaire Unix
- ENUM('value1','value2',...) :
  - Ensemble fixé de valeurs
- SET('value1','value2',...) :
  - Une ou plusieurs parmi

- Et beaucoup d'autres

MySQL – opérations

- Création de bases de données
  - CREATE DATABASE nom_bd
- Suppression de bases de données
  - DROP DATABASE [IF EXISTS] nom_bd
    - IF EXISTS évite une erreur si la base n'existe pas.
- Utilisation d'une base de données
  - USE nom_bd
    - La base spécifiée sera utilisée par défaut.

MySQL – opérations

- CREATE TABLE : permet de créer une nouvelle table dans la base de données courante
  - CREATE TABLE tbl_name [(champ1, ...)]
  - AUTO_INCREMENT : le contenu est incrémenté automatiquement après chaque insertion
  - PRIMARY_KEY : clé d'index primaire unique

```
CREATE TABLE `test` (
  `test_id` INT NOT NULL AUTO_INCREMENT PRIMARY KEY ,
  `test_val` INT NOT NULL)
```

MySQL – opérations

- Suppression d'une Table
  - DROP TABLE tbl_name [, tbl_name, …]
- Modifier la structure d'une table existante
  - ALTER TABLE tbl_name alter_spec [, alter_spec…]

MySQL – opérations

- Insérer de nouveaux enregistrement

```
INSERT INTO article
    (Champ1, Champ2)
    VALUES ('12', 'test');
```

- Remplacer un enregistrement

```
REPLACE INTO article
    (Champ1, Champ2)
    VALUES ('13', 'test');
```

MySQL – opérations

- Modification des valeurs

```
UPDATE table_name
    SET column_name=expr1
    [WHERE section_condition_where]
```

- Suppression d'enregistrements

```
DELETE FROM table_name
    [WHERE definition]
```

MySQL – opérations

- Sélection d'enregistrement

  SELECT [DISTINCT|ALL] expression_de_selection
  FROM tables
  WHERE expression_where
  [GROUP BY col_name, …]
  [HAVING where_definition]
  [ORDER BY [ASC|DESC]]

- Exemples
  - SELECT * FROM article WHERE PrixArt > 50
  - SELECT NumGrArt, AVG(PrixArt) FROM article GROUP BY NumArt

UN EXEMPLE "COMPLET" SIMPLE

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
<head><title>Site de vente en ligne</title></head>

<body>
<div id="container">
  <div id="header">
    <ul id="top_menu">
      <li><a href="index.php">index</a></li>
      <li><a href="rechercher.php">rechercher</a></li>
      <li><a href="commander.php">commander</a></li>
    </ul>
  </div> <!-- /header -->

  <div id="main">

<!– Affichage du catalogue venant de la base de données -->

  </div> <!-- /main -->

  <div id="footer">
    <p>Copyright 2011 &copy; SdVeL. Tous droits réservés</p>
  </div> <!-- /footer -->

</div> <!-- /container -->

</body>
</html>
```

config.php

```php
<?php
session_start();

$user = "root";
$pwd = "";
$host = "127.0.0.1" ;
$bdd = "test";

try {
    $dbh = new PDO('mysql:host='.$host.';dbname='.$bdd, $user, $pwd);
} catch (Exception $e) {
  die('Erreur : ' . $e->getMessage());
}
$menu = array(
  "index.php" => "index",
  "rechercher.php" => "rechercher",
  "commander.php" => "commander"
);
?>
```

header.php

```
<?php include_once('config.php'); ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
    strict.dtd">
<html>


<head><title>Site de vente en ligne</title></head>


<body>
<div id="container">
  <div id="header">
    <ul id="top_menu">
<?php
foreach ($menu as $page_url => $page_title)
  echo "      <li><a href=\"$page_url\">$page_title</a></li>\n";
?>
    </ul>
  </div> <!-- / header -->


  <div id="main">
```

footer.php

```
  </div> <!-- / main -->

  <div id="footer">
    <p>Copyright 2011 &copy; DVD Club. Tous droits réservés</p>
  </div> <!-- / footer -->

</div> <!-- / container -->

</body>
</html>
```

Toutes les autres pages

- Création d'autant de pages que nécessaire :
  - Modification de l'entête ou du pied de page via les fichiers header.php, footer.php
  - Modification des identifiants de connexion via le fichier config.php
  - …
- Reste à modifier les fichiers header/footer et faire une css.

```
<?php include('header.php'); ?>
<!- Affichage du catalogue venant de la base de données -->
<?php
$reponse = $dbh->query('SELECT * FROM `test`');
  if ($reponse->rowCount() > 0) {
    while ($donnees = $reponse->fetch()) {
      echo ...
    }
  } else {
    echo "<li>aucun résultat</li>";
  }
?>
<?php include('footer.php'); ?>
```

Pour aller plus loin

- Il faut ensuite apprendre :
  - Quelques fonctions de base supplémentaires
  - A ne jamais faire confiance au client
  - La gestion des objets en Php
  - Les méthodes de programmation avancées
    - MVC
    - Gestion de templates
    - …