

PHP - deuxième année

Pierre BETTENS
pb@namok.be

ÉPSE - École de Promotion Sociale d'Enghien

09/10/2009

- Cours
 - Exposé oral et manipulations
 - 160 périodes (13h 20m)
 - Réparties en deux modules de 80 périodes (66h 40m)
- Supports
 - Copie des transparents
 - Site web, <http://les1.namok.be>
 - Références
 - Manuel PHP
 - Learning PHP&MySQL, Michele E.Davis & Jon A. Philips, Ed. O'Reilly
- Évaluation
 - Évaluation continue au cours des deux modules
 - Évaluation finale au terme du second module (en 3^e)
 - 50% pour le cours
 - 25% cote artistique par Héléne Masse
 - 25% cote du jury de fin de 3^e

- Organisation
- Préalables
- Introduction à PHP
- Notions de base de données
- Utilisation des formulaires
- Conception de bases de données
- Pérennité des données
- Notions de base de données II
- PHP, plus avant ...
- ...

Préalables

Préalables : : Aspects client-serveur

Préalables : : Architecture web

● Préalables

Aspects client-serveur
Architecture web
Stockage de l'information

- Client Web
 - **Définition** : Un client Web (ou HTTP) est un programme capable de se connecter à un serveur Web. Le client le plus habituel est le *browser* (navigateur internet). Il en existe d'autres :
 - Aspirateur de site (*sujet*)
 - Via les langages de programmation
 - Lecteur de flux RSS/Atom
 - Quelques browsers : MS Explorer, Firefox, Opera, Lynx, ...
- Serveur Web
 - **Définition** : Un serveur web est un programme capable de répondre à des requêtes HTTP.
 - HTTP (Hyper Text Transfert Protocol) est un protocole de communication sur le web
 - Quelques serveurs web :
 - Apache HTTP server, produit (libre) Apache Software Foundation
 - Sun Java System Web Server, produit SUN
 - IIS (Internet Information Server), produit Microsoft

- Technologie HTML
 - Site web **statique**
 - Le serveur transmet les fichiers HTML qu'il détient aux clients qui en font la demande
 - Le fichier HTML (+CSS) est interprété par le client
- Technologie HTML / JavaScript
 - Site web **"animé"**
 - Le serveur transmet les fichiers HTML / JavaScript qu'il détient aux clients qui en font la demande
 - Le fichier HTML / JavaScript est interprété par le client, le JavaScript étant exécuté par le client

Préalables : : Architecture web

Préalables : : Stockage de l'information

Introduction à PHP

- Site web **dynamique**
 - **Motivation** : Volonté de fournir des pages web différentes en fonction du contexte.
 - Évolution de l'ensemble des données ; une liste d'objets par exemple
 - Différentiation des pages en fonction de l'utilisateur
 - ...
 - Le serveur doit faire un travail supplémentaire avant de transmettre les pages HTML
- Technologies
 - CGI, exécution d'un programme du côté serveur ... qui doit faire confiance au programme
 - Utilisation d'un langage de programmation interprété par le serveur ; PHP (Open source), ASP et ASP.NET (Microsoft), JSP (Sun)

- Les fichiers "sources" sont stockés sur le serveur
 - Fichiers HTML, CSS
 - Fichiers PHP, JSP, ...
- Les informations "variables" sont stockées
 - dans un fichier
 - dans une base de données
- Les SGBB (Système de Gestion de Base de Données) les plus courants sont
 - MySQL (open source)
 - PostgreSQL (open source)
 - Oracle (Oracle)
 - DB2 (IBM)
 - SQL Server (Microsoft)

● Introduction à PHP

Mise en œuvre
Mise en place de l'environnement PHP
Comentaires et ;
Les variables
Les constantes, constantes magiques et superglobales
Les expressions et les opérateurs
Les fonctions
Structures de contrôle

Inclure du code PHP dans du code HTML

- Sur un serveur web standard, le fichier doit porter l'extension php (ou php4, php5, ...)
- Le code PHP se trouve entre les balises
 - <?php et ?> ou
 - <script language="php"> et </script> ou encore
 - <? et ? (non recommandé)

Exemple

```
<?php echo date("D, d/m/Y"); ?>
```

- Si ce script est exécuté, il fournit la chaîne en fonction de la date du jour - Thu, 14/08/1971

Exemple (suite)

- Contenu dans du HTML, le fichier aura la forme

```
<html>
...
<p class="result">
  Sous sommes aujourd'hui le <?php echo date("D, d/m/Y") ?>
</p>
</html>
```

- Voir l'exemple [\[browser\]](#) [\[pdf\]](#)

- Notez bien** Nous parlons bien du *fichier source*, pas du fichier interprété et renvoyé par le serveur.

Écrire un fichier complètement en PHP

- On peut envisager d'écrire le tout en PHP

```
<?php
echo "<DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0"
echo " Transitional//EN" ">";
echo "<html>\n";
...
echo "</html>\n";
?>
```

- En général, on aura une approche hybride en fonction de la complexité du problème, du site web à mettre en œuvre, ...
- L'interpréteur PHP ne tient pas compte de ce qu'il trouve entre un *tag fermant* et un *tag ouvrant*

Interpréter le code PHP

- Définition** : Un langage est interprété lorsque les instructions sont traduites et exécutées instruction par instruction
- PHP est **interprété**, pas compilé
- Comment interpréter le code
 - via un **serveur web**
 - via un interpréteur (exemple `php-client`)

Installation de PHP

- Sous Linux
 - Debian way : `apt-get install php5`
 - À voir en fonction de la distribution
 - Remarque** : le paquet `php-cgi` permet un interpréteur en ligne de commande
- Sous windows
 - EasyPHP, WAMP par exemple, permettent de tester ses scripts
 - Ce paquet comprend Apache, PHP et MySQL.
- Sous Mac OS X
 - Paquet MAMP (à vérifier)

Éditer le code

- Utilisation d'un éditeur de code : `gVim`, Notepad++, ...
- Utilisation d'un IDE : `Eclipse`, `Netbeans`, ...
- Utilisation d'un IDE "web" : `Dreamweaver`, ...
- Se documenter
 - Documentation officielle de PHP [\[download page\]](#)
- Notez bien** Être capable d'utiliser la documentation **officielle** d'un outil est un gain de temps appréciable

Semicolon (le point-virgule) ;

- Séparateur d'instructions
- Termine chaque instruction
- Peut-être omis en fin de bloc, le *tag fermant* inclut automatiquement le point-virgule

Les commentaires

- Les commentaires peuvent apparaître sur une ou plusieurs lignes

```
/* Ceci est un commentaire */
// Ceci est un commentaire
# Ceci est un commentaire

/* Ceci est un commentaire
 * sur plusieurs lignes
 */
```

```
<p>This is an <?php # echo 'simple';?> example.</p>
<p>The paragraph above will say "This is an example."/p>
```

- Définition** : Une *variable* représente un endroit où stocker de l'information

- Une variable possède les caractéristiques suivantes

- un *nom (name)* pour la désigner
- un *type (type)* définissant ce qu'elle représente (un nombre ou une chaîne de caractères par exemple)
- une *valeur (value)* qui pourra évoluer au cours du temps
- son *adresse (address)*, l'endroit en mémoire où elle se trouve
- sa *visibilité (scope)* détermine "où" voit (accède) la variable
- sa *durée de vie (lifetime)* la portion de code dans lequel elle existe

nom

- Commence par un **\$**
- Suivi par le nom de la variable *sensible à la casse*
 - commence par une lettre ou underscore `_`
 - suivi par des lettres et/ou des chiffres

Exemples

```
$foo = 1 ;
$FOO = 'Juste';
$_foo = 'Leblanc';
$ps2pd = 'ps2pdf';
$titre = // invalide, commence par un chiffre
```

type

- Le type d'une variable dépend de son contexte
- Les différents types sont
 - boolean
 - integer
 - float (ou double)
 - string
 - array
 - object
 - resource
 - NULL
- Remarque** PHP n'est pas un langage "typé"

type (suite)

- boolean
 - Valeurs, true ou false
 - Insensibles à la casse
 - Une valeur positive est assimilable à true, une valeur nulle à false
- integer
 - Sous-ensemble fini d'éléments de \mathbb{Z}
 - La taille de l'intervalle dépend de la plate-forme (± 32 bits)
 - Littéral en base 10 : notation habituelle
 - Littéral en base 8 : le nombre est précédé d'un 0
 - Littéral en base 16 : le nombre est précédé d'un 0x
 - Exemples

```
<?php
$number = 42 ; // Un entier positif
$number = -5 ; // Un entier négatif
$number = 011 ; // Un entier positif octal
$number = 0x800 ; // Un entier positif hexadécimal
?>
```

type (suite)

- float
 - Sous-ensemble fini et discontinu de \mathbb{R}
 - La taille d'un float dépend de la plate-forme (64 bits)
 - Un float, nombre à virgule flottante permet d'écrire sur un même nombre de bits des nombres grands avec peu de précision et des nombres plus petits avec une plus grande précision
 - Notation décimale, 1.34
 - Notation "scientifique", 1.2E3=1200
- string
 - Représente une chaîne de caractères
 - Un caractère est codé sur un byte (8bits soit 256 caractères différents → pas de support de l'unicode)
 - Trois manières de représenter les littéraux de type string: single quoted, double quoted et heredoc syntax

Page d'accueil

PHP - documentation

18 / 136

Page d'accueil

PHP - documentation

20 / 136

Page d'accueil

PHP - documentation

21 / 136

type (suite)

- string single quoted
 - La chaîne est représentée entre single quote (guillemets simples ou apostrophe)
 - Exemple:


```
$string = "François Pignon" ;
```
 - Pour représenter le ", il faudra l'échapper, c-à-d le faire précéder de \
 - Les variables et les séquences d'échappement ne sont pas étendues
 - On privilégie cette manière de quoting
- string double quoted
 - La chaîne est représentée entre double quote (double guillemets)
 - Exemple


```
$string = "Juste Leblanc" ;
```
 - Pour représenter le ", il faudra l'échapper
 - Les variables et les séquences d'échappement sont étendues

type (suite)

- string heredoc syntax
 - Les trois symboles «<<» sont suivis d'un identifiant et déterminent le début de la chaîne
 - La fin de la chaîne est déterminée par la répétition de l'identifiant
 - L'identifiant de fermeture doit commencer en première colonne
 - Les conventions de nommage sont les mêmes que pour les variables
 - Exemple


```
$mon_poisson = <<<EDQ
Je long d'un siècle ruissseau buvait une colombe,
quand sur l'eau se penchait une fourmis y tombe;
Et dans cet océan l'on eût vu la fourmis
s'efforcer, mais en vain, de regagner la rive.
...
EDQ;
```
 - Les variables et les séquences d'échappement sont étendues

type (suite)

- Arrays
 - Un tableau PHP est un map (carte), ensemble de paires clé-valeur
 - La clé est soit un entier, soit une chaîne (String), la valeur est n'importe quel objet
 - Création grâce à l'instruction array


```
$aArray = array (
    1 => "un",
    2 => "deux" );
```
 - Accès aux éléments


```
$aArray['1'] ;
```
 - Ajout d'éléments


```
$aArray[] = "Trois" ;
echo $aArray[3];
```

Page d'accueil

PHP - documentation

22 / 136

Page d'accueil

PHP - documentation

23 / 136

Page d'accueil

PHP - documentation

24 / 136

type (suite)

- string accéder aux caractères
 - On accède à un caractère en renseignant son indice dans la chaîne à partir de 0
 - Exemples


```
$string = "Be" ;
$char = $string[0] ; // $char vaut 'B'
$string[1] = "o" ; // $string vaut 'Bo'
```
- Les autres types (arrays, object, resource et NULL) seront vus plus tard

visibilité

- Par défaut les variables sont **locales**
- Elles sont visibles dans le bloc dans lequel elles se trouvent
- Pas dans les fonctions définies par l'utilisateur
- Mot-clé global
 - Permet de déclarer une variable comme étant visible partout


```
<?php
global $foo ;
?>
```

- Définition** : Une constante est une variable dont la valeur ne change pas au cours du temps
- Utilisation du mot clé, define ()

```
<?php
define('FOO', "something") ;
?>
```

- Par convention les constantes s'écrivent en majuscule

Page d'accueil

PHP - documentation

25 / 136

Page d'accueil

PHP - documentation

26 / 136

Page d'accueil

PHP - documentation

27 / 136

- Les constantes magiques sont des constantes définies par PHP
- Exemple [\[browser\]](#) [PDF]
- Certaines constantes ont une valeur différentes suivant l'emplacement où elles sont utilisées
 - __LINE__ __FILE__ __DIR__ __FUNCTION__ __CLASS__
 - __METHOD__ __NAMESPACE__

- Les superglobaux (superglobales) sont des variables internes, toujours définies quel que soit le contexte
 - \$GLOBALS référence les variables disponibles dans un contexte global
 - \$_SERVER tableau contenant des informations propres au serveur
 - \$_GET tableau associatif des valeurs passées au script via **get**
 - \$_POST tableau associatif des valeurs passées au script via **post**
 - \$_FILES tableau associatif des informations sur un fichier envoyé au serveur
 - \$_COOKIE tableau associatif des valeurs d'un cookie déjà présent chez le client
 - \$_SESSION tableau associatif des valeurs associées à une **session**
 - \$_REQUEST étroitement lié au superglobaux, GET, POST et COOKIE
 - \$_ENV représente les valeurs des variables d'environnement dans lequel est lancé PHP
- Exemple [\[browser\]](#) [PDF]

- PHP est capable de récupérer des variables passées
 - par le biais d'un formulaire
 - directement dans l'URL
- Il faudra toujours être prudent lors de la récupération de ces valeurs (sécurité)
- En règle générale les valeurs reçues par ce biais ont toutes les chances de ne pas être les valeurs attendues.
- Remarque : Si un variable contient un point, PHP le transformera en " _"

Introduction à PHP :: Les variables externes à PHP :: GET

Introduction à PHP :: Les variables externes à PHP :: POST

Introduction à PHP :: Les expressions

- La méthode "GET" permet le passage des valeurs dans l'URL

Exemple `http://exemple.com/?var=valeur`

```
<?php echo $_GET['var']; ?>
```

- Par prudence, on retire les caractères spéciaux dans ce cas

```
<?php
$maVar = htmlspecialchars($_GET['var']);
#ou
$maVar = htmlentities($_GET['var']);
?>
```

- La méthode "POST" permet le passage de valeurs par le biais d'un formulaire html.

- Le code HTML permettant l'envoi des valeurs

```
<!--
<form name="myForm" type="text" method="post">
  <input type="text" name="var" size="20" />
  <input type="submit" value="Send" />
</form>
-->
```

- Le code PHP permettant leur récupération

```
<?php
echo htmlentities($_POST['var']);
?>
```

- Exemple [\[browser\]](#)

- Définition** : Une expression (au sens large) est quelque chose qui a une valeur

- Exemples d'expressions

- 42
- (3+1)*10+2
- "Hello"
- Si \$foo=5, alors \$foo-2 est une expression
- Plus fort, \$foo=5, est une affectation et aussi une expression

Introduction à PHP :: Les opérateurs

Introduction à PHP :: Les opérateurs

Introduction à PHP :: Les fonctions

- On distingue les opérateurs unaire, binaire et ternaire en fonction du nombre d'opérandes.

- Les opérateurs ont certaines priorités, dans le doute, parenthèse

- Voici quelques opérateurs

- Opérateurs arithmétiques
 - unaire, -
 - binaire, +, -, *, /, %
 - la division est entière si les opérandes sont entières, réelle sinon
- Opérateurs d'incrémentatation
 - post-incrémentatation, décrémentatation, \$i++, \$i--
 - pré-incrémentatation, décrémentatation, ++\$i, --\$i
- Opérateurs logiques
 - && (et), || (ou), XOR, !
- Opérateurs de comparaisons
 - ==, ==~, != (<), !=~, <, >, <=, >=

- Voici quelques opérateurs (suite)

- Opérateurs de comparaisons
 - ==, ==~, != (<), !=~, <, >, <=, >=
- Opérateur de contaténation des string
 - .
- Opérateur d'assignation
 - =, permet d'affecter (assigner) une valeur à une variable
 - Il existe des opérateurs d'"opération/assignation"
 - \$i += 2 est équivalent à \$i = \$i + 2
 - \$s = "foo" est équivalent à \$s = \$s . "foo"
- Opérations sur les tableaux (attention)
 - + , union de tableau
 - aura la taille du plus grand des deux tableaux
 - l'ordre des opérandes a de l'importance
 - ...
- Opérateurs sur les bits
 - unaire, ~
 - binaire, &, |, ^

- Définition** : Une fonction est un bloc de code qui porte un nom, qui peut recevoir des paramètres et retourner une valeur.

- Utilités

- Réutilisation du code
- Lisibilité du code
- Découpage du code (modularité)

- Exemple d'appel

```
<?php
foo(1,"ou");
?>
```

- Exemple de définition

```
<?php
function foo($arg_1, $arg_2, /... / $arg_n) {
  echo "Exemple de fonction.\n";
  return 0;
}
?>
```

Exemples

```
<?php
function somme($op1,$op2) {
    return $op1 + $op2;
}
somme(5,4,2);
?>
```

```
<?php
function display_list($i1, $i2, $i3, $i4, $i5) {
    echo "<ul>";
    echo "<li>$i1</li>\n";
    ...
    echo "</ul>\n";
    return;
}
display_list("Premier item", "Second item", "Trois",
"Quatre", "Cinq");
?>
```

Remarque : Que pensez de second exemple ?

Pierre BERTHOIS

PHP - deuxième année

37 / 136

Pierre BERTHOIS

PHP - deuxième année

38 / 136

Pierre BERTHOIS

PHP - deuxième année

39 / 136

- Une fonction peut être définie partout dans le code
- Pas d'overriding (redéfinition d'une fonction déjà écrite)
- Inutile de définir une fonction avant de l'utiliser
- (Possibilité de définition conditionnelle)
- Passage de paramètres
 - Par valeur
 - Par référence, utilisations de & dans la déclaration de la fonction
- Valeur par défaut de paramètres
- Nombre variable de paramètres (voir les fonctions func_num_args(), func_get_arg() et func_get_args())

```
function foo($arg) {
```

```
function foo($var="valeur") {
```

```
function foo($args) {
    func_num_args();
    func_get_arg();
}
```

Pierre BERTHOIS

PHP - deuxième année

38 / 136

Pierre BERTHOIS

PHP - deuxième année

39 / 136

L'instruction conditionnelle if, else

```
if (condition) {
    instructions
} else {
    instructions
}
```

Exemple

```
<?php
if ( $n > 0 ) {
    echo 'N est strictement positif' ;
} else {
    echo 'N est négatif (ou nul)' ;
}
?>
```

Pierre BERTHOIS

PHP - deuxième année

40 / 136

Pierre BERTHOIS

PHP - deuxième année

41 / 136

Pierre BERTHOIS

PHP - deuxième année

42 / 136

L'instruction conditionnelle if, elseif, else ("if" imbriqués)

```
if (condition) {
    instructions
} elseif (condition) {
    instructions
} else {
    instructions
}
```

Exemple

```
<?php
if ( $n > 0 ) {
    echo 'N est strictement positif' ;
} elseif ( $n == 0 ) {
    echo 'N est nul' ;
} else {
    echo 'N est strictement négatif' ;
}
?>
```

Pierre BERTHOIS

PHP - deuxième année

41 / 136

Pierre BERTHOIS

PHP - deuxième année

42 / 136

L'instruction répétitive do while

```
do {
    instructions
} while (condition) ;
```

- Répéter les instructions tant que la condition est vraie
- Exécution du corps de la boucle - test de la condition

Exemple

```
<?php
$i = 0;
do {
    echo $i;
} while ( $i > 0 );
?>
```

Pierre BERTHOIS

PHP - deuxième année

43 / 136

Pierre BERTHOIS

PHP - deuxième année

44 / 136

Pierre BERTHOIS

PHP - deuxième année

45 / 136

L'instruction répétitive for

```
for (expr1; expr2; expr3)
    instructions
```

- Pour "expr1" jusqu'à "expr2" par pas de "expr3", répéter "instructions"
- "expr1" est évaluée - si "expr2" est vraie le corps de la boucle est exécuté ainsi que "expr3" - ensuite retour à l'évaluation de "expr2" ...

Exemple

```
<?php
for ( $i = 1; $i <= 10; $i++ ) {
    echo $i;
}
?>
```

Pierre BERTHOIS

PHP - deuxième année

44 / 136

Pierre BERTHOIS

PHP - deuxième année

45 / 136

L'instruction conditionnelle if

```
if (condition)
    instructions
```

Exemple

```
<?php
if ( $n > 0 )
    echo "Nombre N est positif" ;
?>
```

- Pour plusieurs instructions, utiliser un bloc ({})

Pierre BERTHOIS

PHP - deuxième année

39 / 136

L'instruction répétitive while

```
while (condition)
    instructions
```

- Tant que la condition est vraie, exécuter les instructions
- Test de la condition - Si la condition est vraie, exécution du corps de la boucle

Exemple

```
<?php
$i = 0 ;
$n = 10 ;
while ( $i < $n ) {
    echo "$i*$n = " . $i*$n . "\n" ;
    $i++ ;
}
?>
```

Pierre BERTHOIS

PHP - deuxième année

42 / 136

L'instruction répétitive foreach

```
foreach (array_expression as $value)
    instructions
```

OU

```
foreach (array_expression as $key => $value)
    instructions
```

- Permet d'itérer sur les valeurs d'un tableau (array)

Exemple

```
<?php
$array = array ('Un', 'Deux', 'Trois' );
foreach ($array as $value) {
    echo $value ;
}
?>
```

Pierre BERTHOIS

PHP - deuxième année

45 / 136

- L'instruction répétitive **foreach** (suite)

- Exemples

```
<?php
$array = array ('Un' => 'Premier élément',
               'Deux' => 'Deuxième élément',
               'Trois' => 'Troisième élément' );
foreach ($array as $key => $value) {
    echo "Key: $key, associated value: $value";
}
?>
```

```
<?php
$array = array ('Un', 'Deux', 'Trois' );
foreach ($array as $key => $value) {
    echo "Key: $key, associated value: $value";
}
?>
```

- Les instructions **break** et **continue**

- L'instruction **break** permet de sortir du corps d'une instruction while, do-while, for, foreach ou switch

- Exemple

```
<?php
$ix = array ( 1, 3, 7, 71, 42, 12, 9);
foreach ( $ix as $i ) {
    if ( $i % 2 == 0 ) break;
    echo $i ;
}
?>
```

- L'instruction **continue** permet de passer directement (sans exécuter le reste du corps de la boucle) à l'itération suivante

- Exemple

- Dans l'exemple précédent remplacer break par continue permet l'affichage de tous les nombres impairs.

- Inclusion de fichiers, **include<_once>**, **require<_once>**

- Permet l'inclusion d'un fichier
- include inclut le fichier
- require inclut le fichier et provoque une erreur s'il n'est pas trouvé
- include_once et require_once veillent à ne faire qu'une seule inclusion
- Exemple

```
<?php
include('header.php');
... (un peu de html) ...
include('footer.php');
?>
```

Notions de base de données

Notions de base de données :: Préalables

Notions de base de données :: Préalables

• Notions de base de données

Préalables
Introduction au SQL
API SQL élémentaire
Quelques bonnes pratiques

- **Définition** : Une **BD** (base de données) permet de stocker une grande quantité d'information.
- **Définition** : Un **SGBD** (système de gestion de base de données) donne les moyens de manipuler une base de données (ajout, suppression, recherche, ...)
- Exemples
 - Répertoire téléphonique
 - Une ligne représente une personne
 - Une colonne représente une "information"; un nom, un prénom, un numéro de téléphone
 - Collection de bandes dessinées

- De manière simpliste, une base de données peut être vue comme un **tableau**

- une **ligne** représente une entrée dans la base de données
 - chaque **colonne** représente un attribut
- BD, fichiers plats (*flat-file*) équivalent à une feuille de calcul

Prénom	Nom	Email
Juste	Leblanc	jeblanc@example.org
Marlène	Sasœur	msasoeur@example.org

Notions de base de données :: Préalables

Notions de base de données :: SQL

Notions de base de données :: SQL

- Les informations que l'on stocke dans une table risquent d'être redondantes
 - Je stocke des types de **bières** et leur **brasserie** associée
 - Une brasserie **brasse** plusieurs bières
 - Changer l'adresse d'une brasserie impose de la changer pour **chaque** bière de la brasserie
- Je répartis l'information dans plusieurs tables qui sont en relation
- SGBD relationnel
 - composée de plusieurs tables
 - les tables peuvent être mises en relation

- **Définition** : Le langage SQL (*Structured Query Language*) permet de manipuler les données d'un SGBD
- SQL définit des types de données
- SQL définit des instructions
- **Remarque** : Dans la suite nous ne donnerons pas la syntaxe complète des ordres SQL.

- L'instruction CREATE DATABASE
 - Permet la création d'une base de données

```
CREATE DATABASE <db_name>;
```

- Exemple

```
CREATE DATABASE brasserie;
```

L'instruction USE

- Permet de préciser avec quelle DB on travaille

```
USE <database_name>;
```

Exemple

```
USE brasserie;
```

L'instruction CREATE TABLE

- Permet la création d'une table

```
CREATE TABLE <table_name> (
  create definition
);
```

Exemple

```
CREATE TABLE biere (
  biere_id INT NOT NULL AUTO_INCREMENT,
  biere_nom VARCHAR(50),
  degre_alcool INT,
  PRIMARY KEY (biere_id)
);
```

L'instruction DESCRIBE

- Permet de visualiser la structure d'une table

```
DESCRIBE <table_name>;
```

Exemple

```
DESCRIBE biere;
```

```
mysql> describe biere;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| biere_id | int(11) | NO | PRI | NULL | auto_increment |
| biere_nom | varchar(50) | YES | | NULL | |
| degre_alcool | int(11) | YES | | NULL | |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)
```

L'instruction SHOW

- Permet de voir les tables, les DB, ...

```
SHOW DATABASES;
```

ou

```
SHOW TABLES;
```

Exemple

```
SHOW TABLES;
```

L'instruction INSERT

- Permet l'insertion de données dans une table de la DB

```
INSERT INTO <table_name> VALUES (<values>);
```

Exemple

```
INSERT INTO biere VALUES (null, "Quintini", 8);
```

- Un SGBD autorise la modification d'une BD sans perte de données

L'instruction ALTER

- Renommer une table

```
ALTER TABLE <table_name> RENAME <table_newname>;
```

- Changer le type d'un attribut

```
ALTER TABLE <table_name> MODIFY <column> <datatype>;
```

- Changer l'ordre des colonnes

```
ALTER TABLE <table_name>
  MODIFY <column> <datatype>
  AFTER <column>;
```

- Ajouter une colonne

```
ALTER TABLE <table_name>
  ADD <column> <datatype>
  [AFTER <column>;]
```

L'instruction ALTER (suite)

- Renommer une colonne

```
ALTER TABLE <table_name>
  CHANGE <column> <column_newname> <datatype>;
```

- Supprimer une colonne

```
ALTER TABLE table DROP COLUMN column;
```

L'instruction DROP

- Supprimer une table

```
DROP TABLE table;
```

Exemple

```
DROP TABLE brasserie;
```

L'instruction SELECT

- Sélectionner des éléments dans une table ou plusieurs

```
SELECT <columns> FROM <tables>
  [WHERE clause]
  [ORDER BY clause]
  [LIKE pattern];
```

- columns indique la liste des colonnes (séparées par des ,)
- WHERE permettra de choisir quelques lignes uniquement
- une clause WHERE peut contenir AND, OR, NOT et ()
- ORDER BY permettra de trier les lignes
- LIKE pattern où
 - % signifie n'importe quel caractère 0..n
 - _ signifie exactement un caractère

● L'instruction SELECT (suite)

● Exemples

```
SELECT * FROM table;
```

```
SELECT biere_nom FROM biere
WHERE biere_nom="Bintink";
```

```
SELECT * FROM biere
WHERE degre_alcool=8
ORDER BY biere_nom ASC ;
```

● L'instruction UPDATE

● Permet de mettre à jour la DB

```
UPDATE <table_name>
SET <col_name=expr>
[WHERE <clause>];
```

● Exemple

```
UPDATE biere
SET degre_alcool=9
WHERE biere_nom="Quintine" ;
```

● L'instruction DELETE

● Permet de supprimer des éléments (des lignes) dans la table

```
DELETE FROM <table_name>
WHERE <col_name=expr> ;
```

● Exemple

```
DELETE FROM biere
WHERE degre_alcool=9 ;
```

● L'instruction GRANT PRIVILEGES

● Ajouter des utilisateurs à la DB

```
GRANT PRIVILEGES ON <database>.<object>
TO <username>@<hostname>
IDENTIFIED BY <password> ;
```

● Exemple

```
GRANT ALL PRIVILEGES ON *.*
TO jleblanc@localhost
IDENTIFIED BY "mySecret" ;
```

● On accède à une base de données MySQL

- via l'utilitaire en ligne de commande `mysql`
- via un interface web `PHPMyAdmin`
- via l'API PHP
- ... d'autres API

● API PHP

- `mysql_connect`
- `mysql_select_db`
- `mysql_query`
- `mysql_fetch_array`
- `mysql_free_result`
- `mysql_close`
- ...

● Exemple

● Connexion à la base de données et sélection de la BD

```
<?php
// Connexion et sélection de la base
$link = mysql_connect(
    'mysql_host', 'mysql_user', 'mysql_password')
or die('Impossible de se connecter : ' . mysql_error());
echo 'Connecté successfully';
mysql_select_db('my_database')
or die('Impossible de sélectionner la base de données');
?>
```

● Exemple (suite)

● Exécution d'une requête MySQL

```
// Exécution des requêtes SQL
$query = "SELECT * FROM my_table";
$result = mysql_query($query)
or die('Échec de la requête : ' . mysql_error());
?>
```

● Exemple (suite)

- Affichage des résultats dans une table. Les résultats sont fournis dans un array

```
// Affichage des résultats en HTML
echo "<table>\n";
while ($line = mysql_fetch_array($result, MYSQL_ASSOC)) {
    echo "\t<tr>\n";
    foreach ($line as $col_value) {
        echo "\t\t<td>$col_value</td>\n";
    }
    echo "\t</tr>\n";
}
echo "</table>\n";
?>
```

● Exemple (suite)

● Rangement ...

```
<?php
// Libération des résultats
mysql_free_result($result);
// Fermeture de la connexion
mysql_close($link);
?>
```

- Pour d'autres fonctions MySQL, consulter Manuel PHP, section MySQL

Utiliser des fonctions

- Ajoute une couche d'abstraction
 - On préférera définir une fonction bas niveau de connection à la DB

```
db_connect($host, $user, $password, $database);

function db_connect($host, $user, $password, $database) {
    $link = mysql_connect($host, $user, $password)
    or die("Connecting problem : " . mysql_error());
    mysql_select_db($database)
    or die("No db selected");
    return $link;
}
```

- Répartit la difficulté
- Accroît la lisibilité

Séparer le code PHP dans plusieurs fichiers

- Par exemple, pour les paramètres de connection à la DB, on aura un fichier `scat_db_conf.php`

```
<?php
$HOST = "localhost";
$USER = "username";
...
>
```

- Il suffira d'ajouter l'instruction suivante dans le(s) fichier(s) utilisant la DB

```
require_once('db_conf.php');
```

- Noter bien : l'extension du nom de fichier est bien `php`
- Tester la valeur de retour d'une fonction `php`

Utilisation de formulaires

- Text boxes
- Text areas
- Checkboxes
- Radio buttons
- Hidden
- Selects

Utilisation de formulaires

Utilisation de formulaires : Text boxes

Utilisation de formulaires : Text boxes

- Comme nous l'avons déjà abordé, un formulaire html permet l'envoi d'informations au serveur
- Un formulaire doit contenir les informations suivantes :
 - la manière de soumettre l'information au serveur, `method`
 - les éléments à envoyer, `input`
 - le script qui prendra en charge les informations transmises au serveur, `action`
- Voir les exemples [\[browser\]](#) [\[pdf\]](#)

Une **text box** a comme type, **text**

- `<input type="text" name="aname" size="23" maxlength="42"/>`

Exemple, un formulaire de recherche

```
Search:  

<form action=""?php echo(htmlentities($_SERVER['PHP_SELF']));?>
  <input type="text" name="search" />
  </input>
  <input type="submit" value="go" />
</form>
```

Utilisation de formulaires : Text areas

Utilisation de formulaires : Checkboxes

Utilisation de formulaires : Checkboxes

Une **text area** n'est pas de type `input` mais **textarea**

- `<textarea name="aname" cols="#cols" rows="#rows"></textarea>`

Exemple, un formulaire de suggestion

```
Your suggestion



<form action=""?scriptname." method="POST">
  <input type="text" name="suggestion" />
  <input type="submit" value="Send" name="button-suggest" />
</form>
```

- Récupérer la valeur dans un script se fait de la même manière qu'avec une **text box**

Une **checkbox** a comme type, **checkbox**

- `<input type="checkbox" name="aname" value="avalue" />`

Exemple, une petite sélection de types de bières

```
Blonde  Brune  Ambrée 


<form action=""?scriptname." method="POST">
  <input type="checkbox" name="b[]" value="blonde" />
  <input type="checkbox" name="b[]" value="brune" />
  <input type="checkbox" name="b[]" value="ambrée" />
  <input type="submit" value="Send" />
</form>
```

Exemple, un formulaire de recherche (suite)

- Pour récupérer les valeurs, on utilise la superglobale `$_POST`

```
$search = htmlentities($_POST['search']);
$scriptname = htmlentities($_SERVER['PHP_SELF']);
if ($search == '') {
  <form action=""?scriptname." method="POST">
    <input type="text" name="search" />
    <input type="submit" value="go" name="button" />
  </form>
} else {
  echo "You search <strong>".$search.</strong>.";
}
```

- Afin de pouvoir récupérer les valeurs checkées, on les place dans un tableau

Exemple, une petite sélection de types de bières (suite)

```
$boxes = $_POST['b'];
echo "Votre choix de bières: \n";
foreach ($boxes as $box) {
  echo "<input checked="" type="checkbox" /> ".$box.\n";
}
echo *</ul>\n";
```

- Une **radio button** a comme type, **radio**

- `<input type="radio" name="aname" value="avalue" />`
- A l'inverse de **checkboxes** les **radio buttons** sont **exclusifs**

- Exemple, ma couleur préférée

```

Rouge  Vert  Bleu 


<form action="" .description=""
method="POST">
<fieldset>
<label>
Rouge <input type="radio" name="couleur" value="rouge"/>
</label>
<label>
...
</fieldset>
<input type="submit" value="Send"/>
</form>

```

- Récupérer la valeur dans un script se fait facilement

Pense BDD

PHP - deuxième année

84 / 136

Pense BDD

PHP - deuxième année

85 / 136

Pense BDD

PHP - deuxième année

84 / 136

- Un champ caché (**hidden**) a comme type, **hidden**

- `<input type="hidden" name="aname" value="avalue"/>`

- Exemple, choix d'une couleur de bière

```

<form action="action.php"
method="POST">
<label>Choix de bière
</label>
<select name="select">
<option>Blonde </option>
<option>Brune </option>
<option>Ambrée </option>
</select>
</label>
<input type="submit" value="Send"/>
</form>

```

Pense BDD

PHP - deuxième année

85 / 136

Pense BDD

PHP - deuxième année

86 / 136

Pense BDD

PHP - deuxième année

87 / 136

- Exemple, choix de plusieurs couleurs de bière

```

<form action="action.php"
method="POST">
<label>Choix de bière
</label>
<select name="select[]" multiple>
<option>Blonde </option>
<option>Brune </option>
<option>Ambrée </option>
</select>
</label>
<input type="submit" value="Send"/>
</form>

```

Pense BDD

PHP - deuxième année

86 / 136

Pense BDD

PHP - deuxième année

87 / 136

Pense BDD

PHP - deuxième année

87 / 136

Conception de bases de données

Préalables

Types de relations

Normalisation

Méthode Meurise (MCD)

Méthode Meurise (MLD)

- La caractéristique principale d'une bd relationnelle est sa capacité d'être répartie en **plusieurs tables** reliées entre elles (les **relations**).

- Relier des tables entre elles se fait en utilisant une **clé (key)** commune à plusieurs tables

- Inclure la clé d'une autre table afin de former un lien, s'appelle une **relation à clé étrangère (foreign key)**

- Exemple, un utilisateur (via un **uid**) associé à un achat

- Il existe plusieurs types de relations entre les tables

- one-to-one**

- Un élément est associé à un seul autre élément
- Exemple, un magasin de livre associe à un acheteur, une adresse de livraison (et cette adresse ne correspond qu'à un acheteur)

- one-to-many**

- Un élément peut-être associé à plusieurs tables
- La clé de la table "one" apparaît dans la table "many"
- Exemple, le format d'un livre. Chaque livre n'a qu'un seul format mais un format donné se retrouve pour quantité de livres

- many-to-many**

- Deux tables peuvent avoir chacune plusieurs clé de l'autre table
- Une table supplémentaire est créé représentant la relation
- Exemple, un acheteur peut acheter plusieurs livres et un (type) livre peut être acheté par plusieurs acheteurs. Il y a création d'une table intermédiaire.

- Définition** La normalisation est une méthode permettant d'organiser les données de manières plus efficace

- Le but est de minimiser la **duplication** des données

- La duplication de données

- occupe de la place
- rend difficile la maintenance
- accroît les risque d'inconsistance de la bd

- Exemple, cas d'un magasin de livre (bookstore) voulant conserver les infos de ses clients

- Utilisation d'une simple table

- Quid d'un changement d'adresse ?
- Quid d'un livre avec plus de deux auteurs ?
- Quid d'un travail d'ajout d'un livre ?

id	Prénom	Le Nom	Adresse	Pages	Pis	Label	ISBN	Pages	Mois
1001	Jean	Jabarc	Paris 1 100	100	Le Livre Pour Les Nuls	978-2-01-010000-1	100	100	100
1002	Jean	Jabarc	Paris 1 100	100	Le Livre Pour Les Nuls	978-2-01-010000-1	100	100	100
1003	Jean	Jabarc	Paris 1 100	100	Le Livre Pour Les Nuls	978-2-01-010000-1	100	100	100

Pense BDD

PHP - deuxième année

88 / 136

Pense BDD

PHP - deuxième année

89 / 136

Pense BDD

PHP - deuxième année

90 / 136

- La normalisation s'opère en trois phases (étapes) appelées chacune **forme**
 - Première forme normale
 - Deuxième forme normale
 - Troisième forme normale
- Ces trois étapes conduisent à une structure de données plus efficace
- Le processus de normalisation passe d'une étape à la suivante dans l'ordre

- Première forme normale
 - Traite de la **redondance au niveau des colonnes (horizontale)**
 - Pas deux colonnes contenant le même type de valeur
 - Pas deux colonnes contenant l'auteur d'un livre
 - Une valeur par colonne
 - Pas une adresse mais une colonne rue, une colonne zipcode, ...
 - La clé primaire doit distinguer les lignes de manière unique

- Première forme normale (exemple)

idLivr	TitreLivre	Last Name	Prénom	ZipCode	City	Titre	Prénom	Titre	Prénom
10001	Le Petit Prince	St Exupéry	Antoine	91000	Evry	Le Petit Prince	St Exupéry	Antoine	91000
10002	Le Petit Prince	St Exupéry	Antoine	91000	Evry	Le Petit Prince	St Exupéry	Antoine	91000
10003	Le Petit Prince	St Exupéry	Antoine	91000	Evry	Le Petit Prince	St Exupéry	Antoine	91000

- Deuxième forme normale
 - Traite de la **redondance au niveau des lignes (verticale)**
 - Identifier les colonnes dont l'information se répète sur plusieurs lignes (ou bien celles qui ne sont pas dépendantes de la clé) et placer ces informations dans leur propre table ... elles seront référencées par une clé.
 - La seconde approche conduit à donner une table aux valeurs dépendantes des acheteurs, à savoir, les livres et les auteurs

- Deuxième forme normale (exemple)

idLivr	Titre	Prénom	Titre	Prénom
1	Le Petit Prince	St Exupéry	Antoine	91000
2	Le Petit Prince	St Exupéry	Antoine	91000
3	Le Petit Prince	St Exupéry	Antoine	91000

- Troisième forme normale
 - Traite de **lien entre les colonnes et la clé primaire**
 - Visé à bien séparer les types d'informations
 - Par exemple certaines parties de l'adresse d'un utilisateur ne sont pas vraiment en relation avec l'utilisateur
 - On peut se contenter d'écrire son "zipcode" et d'associer dans une autre table "zipcode" et "city_id" et dans une autre table "city_id", "city_name" et "state" et ...
 - Risque d'augmenter le coût en nombre de tables par rapport au gain en non-duplication

- Définition** : La méthode Meurise est un ensemble de règles permettant de modéliser une base de données de façon standardisée et méthodique. (Elle a été introduite en France en 1978)
- Décomposition de l'analyse en
 - l'analyse **conceptuelle** et son modèle conceptuel de données (MCD), indépendant du SGBD utilisé et
 - la traduction du modèle conceptuel en un modèle logique, réalisable sur un système de gestion de base de données (MLD)

- Le modèle conceptuel de données (MCD)
 - Cette **modélisation** conduit à la réalisation d'un schéma **entités-associations** (ou entités-relations)
 - Une **entité** est une population d'individus (de "choses") homogènes; des clients, des factures, des sandwiches, ...
 - Une **association** est une liaison entre plusieurs entités; payer (clients, factures), acheter (sandwiches, clients), ...
 - Un **attribut** est une propriété d'une entité ou d'une association qui le (la) caractérise; nom, prénom, ... caractérisent un client
 - Un **identifiant** est un attribut particulier qui sera unique, différent pour chaque individu d'une entité; un numéro ISBN pour un livre, ...
 - La **cardinalité** d'un lien entre une entité et une association précise le nombre de fois (min et max) qu'une entité est concernée par l'association; un client peut commander 1 à n articles
 - 0, n
 - 1, n
 - 1, 1

- Exemple (MCD)



- Le modèle conceptuel de données (MCD) doit être **normalisé** et pour ce faire répondre à 9 règles de normalisation
 - Les entités, toutes les entités qui peuvent être remplacées par une association doivent l'être
 - Les noms, unicité du nom des entités, associations et attributs
 - Les identifiants, chaque entité possède un identifiant
 - Les attributs
 - remplacer les attributs en plusieurs exemplaires en une association
 - les attributs d'une association, ils doivent dépendre directement des identifiants de toutes les entités en association
 - Les associations, élimination des associations .. qui n'en sont pas car les cardinalités sont 1,1 (entraîne la fusion)
 - Les cardinalités, cardinalité minimale 0 ou 1 et maximale 1 ou n (jamais 2,3,4,...)
 - à ces 6 règles ajouter les 3 formes normales

- Le modèle logique de données (MLD)
 - Les données sont organisées en **table** (à chaque entité correspond une table) où chaque colonne caractérise un attribut
 - Les lignes d'une table doivent être uniques ... pour assurer cette unicité, on définit une **clé primaire**
 - Lorsqu'une colonne d'une table référence une autre table, elle utilise sa clé, on parle de **clé étrangère** (foreign key)
 - On représente cette organisation dans un **schéma relationnel**.
 - les tables sont appelées **relations**

- Traduire un MCD en un MLD relationnel, se fait en 5 étapes
 - Chaque entité devient une table et l'identifiant, la clé primaire
 - Une association binaire 1 : n (une des deux cardinalités maximale veut n), disparaît au profit d'une clé étrangère dans la table côté 0,1 ou 1,1 égale à la clé primaire de l'autre table
 - Une association binaire n : m (les deux cardinalités maximales valent n) devient une **table supplémentaire** dont la clé primaire est composée des deux clés étrangères issues des tables de l'association. Les attributs de l'association deviennent des colonnes de la table
 - Une association binaire 1 : 1 (les deux cardinalités maximales valent 1) se transforme comme une association binaire 1 : n mais la clé étrangère doit être **unique**.
 - Une association non binaire (plus de deux entités en relation) devient une **table supplémentaire** dont la clé primaire est composée des clés étrangères issues des tables de l'association. Les attributs de l'association deviennent des colonnes de la table

Exemple (MLD)



- Le modèle physique de données (MPD)
 - C'est une réalisation dans certains SGBD du modèle logique de données
 - En d'autres mots, c'est la traduction en requêtes SQL du schéma relationnel
- Traite de l'optimisation du stockage des données, dans certaines situations, on pourra
 - envisager l'ajout d'un **index** dans la BD
 - envisager de supprimer certaines tables ajoutées lors du passage en 3NF
 - envisager l'ajout d'une certaine redondance afin d'éviter des jointures (voir plus bas) coûteuses
 - Remarque** : Dans ces deux derniers cas ; la BD n'est plus normalisée et le code utilisant doit veiller à sa cohérence

• Pérennité des données

- Préalables
- Sauvegarde
- Restauration
- Export
- Import

- Il ne suffit plus de faire des backup de ses fichiers sources PHP et autres, mais bien des données de la BD
- Rappel, comment faites-vous vos backup ?
- Plusieurs solutions
 - Faire un backup des fichiers de la base de données
 - Mauvaise idée
 - Restauration nécessite la même version du SGBD (installée au même endroit...)
 - Faire un dump de la BD
 - MySQL fournit un fichier contenant les commandes SQL permettant de recréer la BD.

- mysqldump, commande permettant de générer un fichier texte contenant les requêtes SQL de création de la db et des données qu'elle contient
- Possibilité de sauvegarder une table, une base de données ou bien tout
- mysqldump -u <user> -p <objects-to-backup>
 - Envoie le backup sur la sortie standard .. qu'il faudra rediriger dans un fichier
 - mysqldump -u <user> -p <dbname>, une base de données
 - mysqldump -u <user> -p <dbname tablename>, une table de la base de données
 - mysqldump -u <user> -p --all-databases, toutes les base de données
- Exemple
 - mysqldump -u user -p exercices contacts > backup.sql

- mysql permet la restauration des données ... sur base d'un fichier
- mysql -u <user> -p < <backupfile.sql>
- mysql -u <user> -p -D <dbname> < <backupfile.sql>, permet la restauration d'une seule base de données
- Exemple
 - mysql -u user -p -D user < backup.sql

- mysqldump permet l'export des données au format CSV
- mysqldump -u <user> -p --tab=/home/user/backup
 - [-no-create-info] [-no-data]
 - [-fields-terminated-by=","] [-fields-enclosed-by=""]
- --tab précise où sauvegarder l'information (attention à la fois le "user" et "mysq user" écrivent dans ce répertoire)
- --no-create-info ne crée pas le fichier de structure de la table
- --no-data ne crée pas le fichier de données
- --fields-terminated-by précise le séparateur de champs (par défaut TAB)
- --fields-enclosed-by précise un éventuel délimiteur de chaînes (par défaut rien)

- mysqlimport permet d'importer des données (au format CSV) dans une table de la base de données
- mysqlimport -u user -p [--fields-terminated-by=","] [-fields-enclosed-by=""] databasename tablename.txt
 - Le nom du fichier détermine le nom de la table
 - La table doit exister, cette commande sert à la remplir
 - Il existe d'autres paramètres, comme --fields-enclosed-by

Notions de base de données II

Jointures
 Jointures naturelles
 Jointures internes
 Jointures externes
 Jointures croisées
 Conditions de jointures

Notions de base de données II : : Jointures

- **Définition** : Une *jointure* permet d'exécuter une requête sur plusieurs tables (et de profiter pleinement de l'aspect relationnel des données).
- La majorité des jointures impose l'égalité d'une colonne d'une table à une colonne d'une autre table, on parle alors de *jointure naturelle*
 - Exemple, deux tables adresses et users contiennent un attribut addressid qui sera utilisé dans la jointure

Notions de base de données II : : Jointures

- Produit cartésien **sans** critère de jointure (*bad idea*)

```
SELECT address, city, lastname, firstname
FROM adresses, users ;
```

- Va générer "nombre de lignes table adresses" x "nombre de lignes table users"

- Produit cartésien **avec** critère de jointure

```
SELECT address, city, lastname, firstname
FROM adresses, users
WHERE adresses.addressid = users.addressid ;
```

- Ou bien, si l'on renomme les tables

```
SELECT address, city, lastname, firstname
FROM adresses [AS] a, users [AS] u
WHERE a.addressid = u.addressid ;
```

Notions de base de données II : : Jointures

- Produit cartésien **avec** critère de jointure (suite)

- On peut ajouter des conditions supplémentaires dans la clause where

```
SELECT address, city, lastname, firstname
FROM adresses AS a, users AS u
WHERE a.addressid = u.addressid
AND city = 'My city'
```

```
SELECT address, city, lastname, firstname
FROM adresses AS a, users AS u
WHERE a.addressid = u.addressid
AND city LIKE 'My*'
```

- On peut également utiliser les alias dans la partie SELECT de la requête (pour accéder un champ présent dans les deux tables)

```
SELECT address, city, lastname, firstname, a.addressid
FROM adresses AS a, users AS u
WHERE a.addressid = u.addressid
AND city = 'My city'
```

Notions de base de données II : : Jointures : : JOIN

- La clause JOIN

- Utilisation d'une clause spécifique, **JOIN**, aux jointures plutôt que dans la clause WHERE

- Jointure naturelle

- Cette jointure joint deux tables de manière naturelle, c'est-à-dire sur base de l'attribut de même nom dans les deux tables

```
SELECT [DISTINCT ou ALL] <* ou liste de colonnes>
FROM <table gauche>
NATURAL JOIN <table droite>
[USING (colonne1 [, colonne2 ...])]
-- USING permet de définir la colonne concernée par la jointure
```

- Exemple

```
SELECT firstname, lastname, address, city
FROM adresses
NATURAL JOIN users
```

- Jointure interne

- C'est la jointure par défaut, d'ailleurs le mot clé INNER est optionnel

```
SELECT [DISTINCT ou ALL] <* ou liste de colonnes>
FROM <table gauche>
[INNER] JOIN <table droite>
ON <condition de jointure>
```

- Exemple

```
SELECT firstname, lastname, address, city
FROM adresses a
JOIN users u
ON a.addressid = u.addressid
```

Jointure externe

- Une jointure externe permet de recevoir plus d'informations qu'avec d'autres jointures. Les jointures externes donnent aussi les lignes de la table gauche n'ayant pas de valeur dans la table droite ... ce que ne fait pas une jointure interne

```
SELECT [DISTINCT ou ALL] <*> ou liste de colonnes>
FROM <table gauche>
LEFT | RIGHT | FULL [OUTER] JOIN <table droite>
ON <condition de jointure>
```

Exemple

```
SELECT first_name, last_name, address, city
FROM addresses a
RIGHT JOIN users u
ON a.addressid = u.addressid
```

Jointure croisée

- Une jointure croisée fait simplement le produit cartésien des tables en présence

```
SELECT [DISTINCT ou ALL] <*> ou liste de colonnes>
FROM <table gauche>
CROSS JOIN <table droite>
```

Exemple

```
SELECT isbn, titre, format.format
FROM books CROSS JOIN format ;
```

- Toutes les jointures ne sont pas des "equi-jointures" (basée sur l'égalité de deux colonnes)

On distingue

- >, supérieur
- >=, supérieur ou égal
- <, inférieur
- <=, inférieur ou égal
- <>, différent de
- IN, dans un ensemble
- LIKE, correspondance partielle (en combinaison avec les symboles ? et %)
- BETWEEN ... AND ..., entre deux valeurs
- EXISTS dans une table

- Une jointure peut être faite sur la même table
 - todo ajouter un exemple

PHP, plus avant ...

Cookie
Sessions
Authentication
...

- Définition** : Un **cookie** (témoignage) est une information transmise par le serveur à un client que ce dernier restitue à ce même serveur à chaque requête
- Les cookies sont utilisés pour fournir une information **personnalisée** à un client
- Les cookies sont stockés dans un (Firefox, Opéra, Safari) ou plusieurs (IE Explorer) fichiers textes.
 - Sous Firefox et Co, `~/mozilla/firefox/<profilname>/cookies.txt`
 - Sous IE Explorer C : TODO
- Contraintes**
 - Leur nombre total est limité (en fonction du navigateur ~300);
 - La taille maximale d'un cookie est de 4 ko;
 - Le nombre maximum de cookies par domaine est limité (~50).

Placer un cookie

```
setcookie (<name>
[, <value>, <expire>, <path>, <domain>, <secure>])
```

- name, le nom du cookie
- value, la valeur du cookie
- expire, un "timestamp unix" sinon expire à la fermeture du browser
- path, le chemin du serveur à partir d'où on peut accéder au cookie
- domain, permet de limiter l'accès à un sous domaine (`http://sub.example.org`)
- secure, si 1, https, si 0 (par défaut http(s))

Lire un cookie

```
$_COOKIE["<cookieName>"]
```

Détruire un cookie

- Le client peut les détruire (et les lire) via son navigateur ou le (ou les) fichiers dans lequel ils sont stockés
- Le serveur peut
 - redéfinir le cookie (de même nom) sans valeur ou
 - fixer la date d'expiration dans le passé

```
setcookie("monCookie", "", Time()-1);
```

De quoi faut-il se méfier ?

- Certains utilisateurs refusent les cookies
- "Jadis" les cookies pouvaient être utilisés pour suivre les internautes (principalement via les banques de publicité)
- Deux sites indépendants partagent une image d'un site tiers, si ce site peut placer des cookies, il peut tracer l'internaute



- Par défaut les pages d'un site sont indépendantes, le serveur ne conserve pas de traces du passage d'un internaute d'une page à l'autre, si ceci peut-être gênant, les sessions sont la solution.
- Définition** : Une session permet de conserver de l'information "cliente" pendant la visite d'un internaute
 - Une session peut sauvegarder de l'information dans plusieurs variables
 - Une session prend fin lorsque l'on ferme le browser (ou si l'on détruit explicitement la session)
 - Le serveur peut conserver cette trace en assignant un identifiant de session (SID, Session ID) en début de session.

- SID, Identifiant de Session**
- Le client envoie le SID à chaque demande de page
 - le SID est stocké dans un cookie ou
 - dans l'URL si le client n'accepte pas les cookies

- Commencer une session

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

- L'appel à cette fonction se fait avant tout envoi de *header* ou autre *output*
- Utiliser les variables de session
 - Utilisation de la *superglobals* `$_SESSION`
 - Assigner la valeur *jeblanc* à la variable *username*

```
$_SESSION['username'] = 'jeblanc';
```

- Utiliser les variables de session (suite)

- Exemple

```
<?php
// Page 1
start_session();
$_SESSION['number']=42;
echo "<e head=page0.php>Go to page 2</a>";
?>
```

```
<?php
// Page 2
start_session();
echo "the answer is " . $_SESSION['number'];
?>
```

- Terminer une session

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

- Remarque** : Même si l'on détruit la session, les variables de session restent disponibles jusqu'à la fin du script
- Remarques**
 - Il est possible de donner un *timeout* à la session (dans un fichier `.htaccess`)
 - Il est également possible de stocker les variables de sessions dans une DB plutôt que dans un fichier temporaire
 - Les sessions et les cookies permettent de mettre en œuvre un accès restreint à certaines pages d'un site

- Avec les cookies et les sessions, il est possible de faire
 - une demande d'authentification
 - une vérification dans une base de données
 - et de sauvegarder dans une session les paramètres d'un utilisateur

- PHP peut faire une demande d'authentification auprès du serveur Apache
- Pour ce faire, envoyer à Apache, des *headers* de demande d'authentification

```
<?php
if ( !isset($_SERVER['PHP_AUTH_USER']) )
{
  if ( !isset($_SERVER['PHP_AUTH_PW']) ) {
    header('WWW-Authenticate: Basic realm="Member Area"');
    header("HTTP/1.0 401 Unauthorized");
    echo "Please login with a valid username and password.";
    exit;
  }
  else {
    echo "You entered a username of: " . $_SERVER['PHP_AUTH_USER'];
    echo " and a password of: " . $_SERVER['PHP_AUTH_PW'] . ". ";
  }
}
?>
```

- ... et contrôler ensuite que le couple *login/password* est valide ... dans une DB par exemple

Conclusion

Points non abordés

Crédits

- PHP orienté objet (prévu pour la troisième année)
 - Concepts généraux
 - Authentification OO
 - Accès aux bases de données OO
- PHP
 - Mécanisme d'exception
 - Variables dynamiques
 - Notions de références
 - Le mot clé `static`
- SQL
 - Les transactions
 - Les index

■ Les slides son générés grâce aux logiciels (libres) suivants

- ◆ FreeMind <http://freemind.sourceforge.net>
- ◆ L^AT_EX <http://latex-project.org>
- ◆ Plugin Beamer pour L^AT_EX

■ Références

- ◆ Manuel PHP
- ◆ *Learning PHP&MySQL*, Michele E.Davis & Jon A. Philips, Ed. O'Reilly