Partie 4 : PHP 5

Plan

Plan

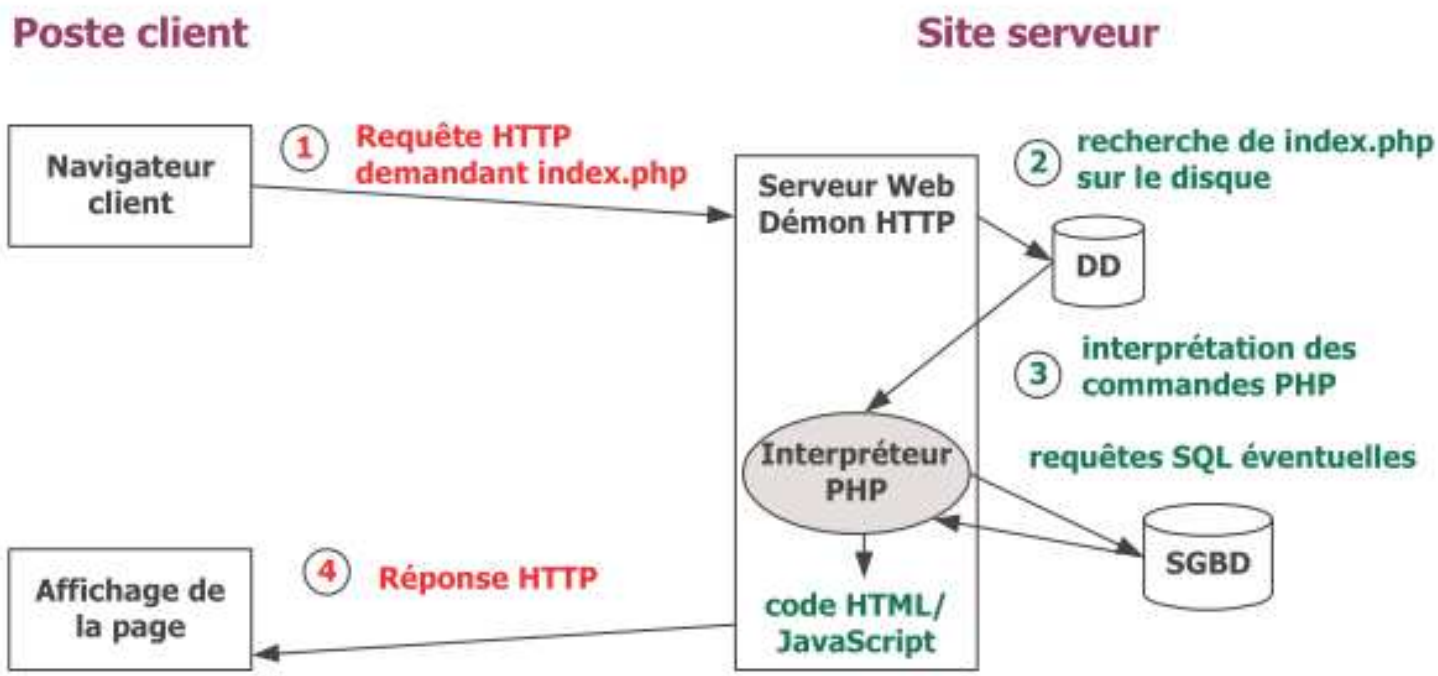
PHP

PHP

➢PHP est un langage de script côté serveur

- ➢ Langage interprété
- ➢ S'exécute coté serveur
- ➢ Le code est intégré au code source de la page HTML
- ➢ Permet de rendre les pages HTML Dynamique

PHP


Poste client
Site serveur
Navigateur client
1
Requête HTTP demandant index.php
Serveur Web Démon HTTP
2
recherche de index.php sur le disque
DD
3
interprétation des commandes PHP
Interpréteur PHP
requêtes SQL éventuelles
SGBD
Affichage de la page
4
Réponse HTTP
code HTML/ JavaScript

PHP

Intégration PHP et HTML

Principe

Les scripts PHP sont généralement intégrés dans le code d'un document HTML

L'intégration nécessite l'utilisation de balises

Avec le style php: <?php ligne de code PHP ?>

avec le style des ASP : <% ligne de code ASP %>

Forme d'une page PHP

Intégration directe

```
< ?php
        //ligne de code PHP
        ?>
        <html>
        <head> <title> Mon script PHP </title> </head>
        <body>
        //ligne de code HTML
        < ?php
        //ligne de code PHP
        ?>
        //ligne de code HTML
        ....
        </body> </html>
```

Forme d'une page PHP

Intégration « indirecte »

Inclure un fichier PHP dans un fichier HTML : include()

Fichier Prinipal

```
<html>
<head>
<title> Fichier d'appel </title>
</head>
<body>
<?php
$salut = " BONJOUR" ;
include "information.inc" ;
?>
</body>
</html>
```

Fichier à inclure : information.inc

```
<?php
$chaine=$salut." ,C'est PHP";
echo "  <table border= \"3"
 <tr> <td width = " 100%\ " >
  <h2> $chaine</h2>
  </td> </tr></table> ";
?>
```

Fonctionnalités de base de PHP

- Commentaires : /* Bloc de commentaires */
  // Ligne de commentaire
- Insensible à la casse pour les **fonctions** et **pas pour les noms des variables**
- Toutes les variables ont un nom précédé par **$**
- Variables non typées à la déclaration (l'affectation détermine le type de la variable)

Fonctionnalités de base de PHP

Les Constantes

- Une constante est un identifiant (un nom) qui représente une valeur simple.
- Une constante ne peut jamais être modifiée durant l'exécution du script.
- Par convention, les constantes sont toujours en majuscules.
- Pour définire une constante hors dans un script et hors d'une classe PHP on utilise la syntaxe suivante :

```
define("nomDeLaCostante",     "valeurDeLaConstante");
```

- Exemple :

```
define("jeSuisUneConstante",     "bonjour");
```

Fonctionnalités de base de PHP

Les Variables

- Une variable en PHP est représentée par un signe dollar "$" suivi du nom de la variable. Le nom est sensible à la casse.

- Une variable dynamique est une variable qui permet d'avoir un nom dynamique d'une variable. Elle prend la valeur d'une variable et l'utilise comme nom d'une autre variable.

- Pour déclarer une variable dynamique on utilise $$.

- Exemple :

```
$ndv = 'varDyn'
$$ndv = 'contenuVarDyn';
```

Ici nous avons deux variables $ndv qui contient la chaine varDyn et $varDyn qui contient la chaine contenuVarDyn

Fonctionnalités de base de PHP

Portée des Variables

- Variable locale

Visible uniquement à l'intérieur d'un contexte d'utilisation

- Variable globale

Visible dans tout le script
Accessible localement avec l'instruction global ou avec la variable globale $GLOBALS

Exemple :

```php
<?php
$globalVar1 = 1;
$globalVar2 = 2;
function somme() {
  global $ globalVar1, $ globalVar2;
  $ globalVar2  = $ globalVar1  +
  $globalVar2;
}
somme();
echo $ globalVar2;
```

```php
<?php
$globalVar1  = 1;
$globalVar2  = 2;
function somme() {
$GLOBALS['globalVar2'] =
$GLOBALS['globalVar1'] +
$GLOBALS['globalVar2'];
}
somme();
echo $ globalVar2;
?>
```

Fonctionnalités de base de PHP

Les Variables

- Fonctions de vérifications de variables

floatval() : Convertit une chaîne en nombre à virgule flottante
empty() : Détermine si une variable est vide
gettype() : Retourne le type de la variable (boolean, integer, double(float), string, array, object, null, unkown type)
intval() : Retourne la valeur numérique entière équivalente d'une variable
isType() : is_array() , is_bool(), is_double(), is_float(), is_int(), is_integer, is_long(), is_object(), is_real(), is_numeric(),is_string() vérifie si la variable est de type Type.
Isset() : Détermine si une variable est définie et est différente de NULL
settype(var, newType) :Affecte un type à une variable
strval() : Récupère la valeur d'une variable, au format chaîne
unset() : Détruit une variable
var_dump() : Affiche les informations d'une variable

- Affectation par valeur et par référence

Affectation par valeur : $b=$a
Affectation par (référence) variable : $c = &$a

Fonctionnalités de base de PHP

Les Variables superglobales

- Variables d'environnement Client

Variable	Description
$_SERVER["HTTP_HOST"]	Nom d'hôte de la machine du client (associée à l'adresse IP)
$_SERVER["HTTP_REFERER"]	URL de la page qui a appelé le script PHP
$_SERVER["HTTP_ACCEPT_LANGUAGE"]	Langue utilisée par le serveur
$_SERVER["CONTENT_TYPE"]	Type de données contenu présent dans le corps de la requête. Il s'agit du type MIME des données
$_SERVER["REMOTE_ADDR"]	L'adresse IP du client appelant le script
$_SERVER["PHP_SELF"]	Nom du script PHP

Fonctionnalités de base de PHP

- Variables d'environnement Serveur

Variable	Description
$_SERVER["SERVER_NAME"]	Le nom du serveur
$_SERVER["HTTP_HOST"]	Nom de domaine du serveur
$_SERVER["SERVER_ADDR"]	Adresse IP du serveur
$_SERVER["SERVER_PROTOCOL"]	Nom et version du protocole utilisé pour envoyer la requête au script PHP
$_SERVER["DATE_GMT"]	Date actuelle au format GMT
$_SERVER["DATE_LOCAL"]	Date actuelle au format local
$_SERVER["$DOCUMENT_ROOT"]	Racine des documents Web sur le serveur

phpinfo(***INFO_VARIABLES***); permet d'afficher les variables d'environnement

Fonctionnalités de base de PHP

Les chaines de caractères

- Une chaine de caractère est une série de caractères.

Quelques fonctions prédéfinies pour la gestion de chaines de caractères

Pour les détails sur ces fonctions et d'autres fonctions de chaines consultez le Manuel PHP : http://php.net/manual/fr/ref.strings.php

chr — Retourne un caractère à partir de son code ASCII
count_chars — Retourne des statistiques sur les caractères utilisés dans une chaîne
echo — Affiche une chaîne de caractères
explode — Coupe une chaîne en segments
fprintf — Écrit une chaîne formatée dans un flux
htmlentities — Convertit tous les caractères éligibles en entités HTML
htmlspecialchars_decode — Convertit les entités HTML spéciales en caractères
htmlspecialchars — Convertit les caractères spéciaux en entités HTML
implode — Rassemble les éléments d'un tableau en une chaîne
lcfirst — Met le premier caractère en minuscule
ltrim — Supprime les espaces (ou d'autres caractères) de début de chaîne
money_format — Met un nombre au format monétaire
nl2br — Insère un retour à la ligne HTML à chaque nouvelle ligne
ord — Retourne le code ASCII d'un caractère
print — Affiche une chaîne de caractères

Fonctionnalités de base de PHP

Les chaines de caractères

printf — Affiche une chaîne de caractères formatée
rtrim — Supprime les espaces (ou d'autres caractères) de fin de chaîne
sprintf — Retourne une chaîne formatée
sscanf — Analyse une chaîne à l'aide d'un format
str_getcsv — Analyse une chaîne de caractères CSV dans un tableau
str_repeat — Répète une chaîne
str_replace — Remplace toutes les occurrences dans une chaîne
str_shuffle — Mélange les caractères d'une chaîne de caractères
str_split — Convertit une chaîne de caractères en tableau
str_word_count — Compte le nombre de mots utilisés dans une chaîne
strcmp — Comparaison binaire de chaînes
strip_tags — Supprime les balises HTML et PHP d'une chaîne
stripos — Recherche la position de la première occurrence dans une chaîne, sans tenir compte de la casse
stripslashes — Supprime les antislashs d'une chaîne
strstr — Trouve la première occurrence dans une chaîne
stristr — Version insensible à la casse de strstr
strlen — Calcule la taille d'une chaîne
strncmp — Comparaison binaire des n premiers caractères

Fonctionnalités de base de PHP

Les chaines de caractères

strpos — Cherche la position de la première occurrence dans une chaîne
strrchr — Trouve la dernière occurrence d'un caractère dans une chaîne
strrev — Inverse une chaîne
strripos — Cherche la position de la dernière occurrence d'une chaîne contenue dans une autre, de façon insensible à la casse
strrpos — Cherche la position de la dernière occurrence d'une sous-chaîne dans une chaîne
strtok — Coupe une chaîne en segments
strtolower — Renvoie une chaîne en minuscules
strtoupper — Renvoie une chaîne en majuscules
strtr — Remplace des caractères dans une chaîne
substr_compare — Compare deux chaînes depuis un offset jusqu'à une longueur en caractères
substr_count — Compte le nombre d'occurrences de segments dans une chaîne
substr_replace — Remplace un segment dans une chaîne
substr — Retourne un segment de chaîne
trim — Supprime les espaces (ou d'autres caractères) en début et fin de chaîne
ucfirst — Met le premier caractère en majuscule
ucwords — Met en majuscule la première lettre de tous les mots

Fonctionnalités de base de PHP

Les tableaux

- Un tableau est une succession d'éléments de différents types
- Un tableau est crée en utilisant la fonction array()
- Les éléments d'un tableau peuvent pointer vers d'autres tableaux
- L'index d'un tableau en PHP commence de 0
- On ne définit pas la taille du tableau
- La fonction count() pour avoir le nombre d'éléments d'un tableau

Fonctionnalités de base de PHP

Les tableaux

- Tableau indicés : Les éléments sont accessible par leur index qui commence par 0.

Exemple

```
$cartes =array(1,2,3,4,5,6,7,8,9,10,11,12,13);
```

Parcours

```
echo "Affichage avec for : <br>";

for ($i=0;$i<count($cartes);$i++){
   echo "carte de valeur : ". $cartes[$i]. "<br>";
}

echo "Affichage avec foreach : <br>";

foreach($cartes as $carte){
   echo "carte de valeur : ". $carte. "<br>";
}
```

Fonctionnalités de base de PHP

- Tableau associatif : L'indice de chaque élément est une chaine de caractère.

Exemple

```
$cartes = array(
« As »=>1, « Dous »=>2, « Tris »=>3, « Quatro »=>4, « Chinka »=>5,
 « six »=>6, « Sept »=>7, « huit »=>8, « neuf »=>9, « Dix »=>10, « Valet »=>11,
« Dame »=>12, « Roi »=>13,
);
```

Parcours

```
foreach($cartes as $indCarte => $Carte) {
   echo "La carte ". $indCarte . " de valeur : " . $Carte . "<br>";
 }
```

http://php.net/manual/fr/language.types.array.php

Fonctionnalités de base de PHP

Quelques fonctions sur Les tableaux

- $tableau = array_count_values($variable) retourne un tableau comptant le nombre d'occurrences des valeurs d'un tableau.
- $tableau = array_diff($var_1, $var_2, ..., $var_N) retourne dans un tableau contenant les valeurs différentes entre deux ou plusieurs tableaux.
- $tableau = array_intersect($var_1, $var_2, ..., $var_N) retourne un tableau contenant les enregistrements communs aux tableaux entrés en argument.
- $tableau = array_merge($var_1, $var_2, ..., $var_N) enchaîne des tableaux entrés en argument afin d'en retourner un unique.

Fonctionnalités de base de PHP

Quelques fonctions sur Les tableaux

- $tableau = array_merge_recursive($var_1, $var_2, ..., $var_N) enchaîne des tableaux en conservant l'ordre des éléments dans le tableau résultant.

- sort($var) : tri les valeurs du tableau selon le code ASCII
  Le tableau initial est modifié et non récupérables dans son ordre original
  Pour les tableaux associatifs les clés seront perdues et remplacées par un indice créé après le tri.

- rsort ($var) tri en ordre inverse des codes ASCII.

- asort ($var) trie également les valeurs selon le critère des codes ASCII, mais en préservant les clés pour les tableaux associatifs

- arsort ($var) la même action mais en ordre inverse des codes ASCII

- natcasesort ($var) effectue un tri dans l'ordre alphabétique non ASCII (« a » est avant « z » et « 10 » est après « 9 »)

Fonctionnalités de base de PHP

Les structure conditionnelles

- L'instruction if

  if (condition réalisée) { liste d'instructions }

- L'instruction if ... Else

  if (condition réalisée)
  {
  liste d'instructions
  }
  else {
  autre série d'instructions
  }

- L'instruction if ... elseif ... Else

  if (condition réalisée) {liste d'instructions}
  elseif (autre condition) {autre série d'instructions }
  else (dernière condition réalisée) { série d'instructions }

- Opérateur ternaire

  (condition) ? instruction si vrai : instruction si faux

L'instruction switch

```
switch (Variable) {
case Valeur1: Liste d'instructions break;
case Valeur1: Liste d'instructions break;
case Valeurs...: Liste d'instructions break;
default: Liste d'instructions break;
}
```

Fonctionnalités de base de PHP

Les structure itératives

- La boucle for
  - for ($i=1; $i<6; $i++) { echo "$i
"; }
- La boucle while
  - While(condition) {bloc d'instructions ;}
- La boucle do…while
  - do {
    bloc d'instructions ;
    }while(condition) ;
- La boucle foreach (PHP4)
  - Foreach ($tableau as $valeur) {insts utilisant $valeur ;}

Fonctionnalités de base de PHP

Déclaration et appel d'une fonction

```
Function nom_fonction($arg1, $arg2, …$argn)
{
   déclaration des variables ;
   bloc d'instructions ;
  // en cas de retour de valeur
   return $resultat ;
}
```

Fonction avec nombre d'arguments inconnu

- **func_num_args()** : fournit le nombre d'arguments qui ont été passés lors de l'appel de la fonction
- **func_get_arg($i)** : retourne la valeur de la variable située à la position $i dans la liste des arguments passés en paramètres.
  - Ces arguments sont numérotés à partir de 0

Fonctionnalités de base de PHP

- Exemple fonction avec nombre d'arguments inconnu :

```
<?php
function produit()
{
$nbarg = func_num_args() ;
$prod=1 ;
for ($i=0 ; $i <$nbarg ; $i++)
{
   $prod *= func_get_arg($i) ;
}
return $prod;
}
echo "le  produit est : ", produit (2, 7, 82, 8, 11, 2016), "<br />" ;
?>
```

Fonctionnalités de base de PHP

- Passage de paramètre par référence
  - Pour passer une variable par référence, il faut que son nom soit précédé du symbole & (exemple &$a) lors de la définition.

```
<?
        function sommeProduit(&$som, &$prod,$x,$y)
        {
           $som=$x+$y;
           $prod=$x*$y;
        }
        $s=0;
        $p=0;
        sommeProduit($s,$p,5,2);

        echo " somme = $s et prod = $p";
?>
```

- L'appel récursif
  - PHP admet les appels récursifs de fonctions

Fonctionnalités de base de PHP

Appel dynamique de fonctions

- Exécuter une fonction dont le nom n'est pas forcément connu à l'avance par le programmeur du script
- L'appel dynamique d'une fonction s'effectue en suivant le nom d'une variable contenant le nom de la fonction par des parenthèses

```php
<?php

 $datejour = getdate();
$heure = $datejour["hours"];
$minute=$datejour["minutes"];
function bonjour(){
    global $heure;
    global $minute;
    echo "<b> BONJOUR A VOUS IL EST : $heure H $minute </b> ";
}
function bonsoir (){
    global $heure;
    global $minute;
    echo "<b> BONSOIR A VOUS IL EST : $heure H $minute <br  />";
}
if ($heure <= 17)  {
    $salut = "bonjour";
    }
else $salut="bonsoir";
//appel dynamique de la fonction
$salut();
?>
```

Fonctionnalités de base de PHP

Les sessions

- Mécanisme permettant de mettre en relation les différentes requêtes du même client sur une période de temps donnée.
- Stocké côté serveur.
- Les sessions permettent de conserver des informations relatives à un utilisateur lors de son parcours sur un site web
- Des données spécifiques à un visiteur pourront être transmises de page en page afin d'adapter personnellement les réponses d'une application PHP
- Chaque session est identifiée par un numéro d'identification dénommé identifiant de session (SID)

Fonctionnalités de base de PHP

- Fonctionnement
  - Un répertoire est créé sur le **serveur** à l'emplacement désigné par le fichier de configuration **php.ini**, afin de recueillir les données de la nouvelle session.

session.save_path = "c:/wamp/tmp";

- Le fichier php.ini peut également préciser la durée de vie d'une session en seconde par **session.gc_maxlifetime**

session.gc_maxlifetime = 1800

Fonctionnalités de base de PHP

- Utilité
  - Sécurisé l'accès à vos pages
    - Conserver le user authentifié
    - Restreindre certaines fonctionnalités à des rôles particulier
    - Passer un « token » pour vérifier qu'on suit un même process.
  - Conserver des données le long de la visite d'un utilisateur
    - Ces coordonnées
    - Le contenu d'un panier pour un site de e-commerce

Fonctionnalités de base de PHP

Les sessions

- Afin de créer une session on utilise la fonction **session_start()**
- Cette fonction doit être appelée à chaque début de page **avant le code HTML.**
- Afin de fermer une session on utilise la fonction **session_destroy()**
- Pour ajouter une variable de session on peut :
  - utiliser **la méthode session_register(nomVariable) (Déconseillé)**
  - Ou directement en utilisant la variable **superglobale $_session**
- **Pour supprimer les variables de sessions utiliser la fonction session_unset()**

Fonctionnalités de base de PHP

Les cookies

- Un cookie est un fichier texte enregistré côté client.
- Il permet d'enregistrer des informations utiles sur et pour le client qui sont généralement utilisées pour ses prochaines visites
- Pour des raisons de sécurité, les cookies ne peuvent être lus que par les pages du serveur créateur du cookie en question.
- La date d'expiration des cookies est définie par le serveur web qui les a crées.
- Les cookies disponibles sont importés par PHP sous forme de variables identifiées sous les noms utilisés par ces cookies
- La variable globale du serveur $_COOKIES enregistre tous les cookies qui ont été définis

Exemple d'utilisation

- Mémorisation des paniers dans les applications d'e-commerce pour une prochaine visite
- Identification des utilisateurs
- Des pages web individualisées
- Afficher des menus personnalisés
- Afficher des pages adaptées aux utilisateurs en fonction de leurs précédents visites

Ecrire un cookie

- bool setcookie (string $name, string $value, int $expire = , string $path , string $domain , bool $secure = false , bool $httponly = false)
- Tout les arguments sauf $name ne sont pas obligatoire
- $name : nom du cookie
- $value : contenu du cookie, n'y stocker pas de mot de passes ou des propriétés critiques ou importantes
- $expire : Le temps après lequel le cookie expire. C'est un timestamp Unix, donc, ce sera un nombre de secondes depuis l'époque Unix (1 Janvier 1970). Il faut donc fixer cette valeur à l'aide de la fonction time() qui permet d'avoir le timestamp actuel en y ajoutant le nombre de secondes après lequel on veut que le cookie expire.

  *Exemple : time()+60*60*24*365* fera expirer le cookie dans 1 an.

Accéder à un cookie

- Pour accéder à un cookie, il faut se souvenir qu'il est stocké dans la variable super globale $_COOKIES ce qui fait qu'on lui accède comme n'importe quel variable dans un tableau associatif via son nom.

- Exemple $_COOKIES['nom'] permet d'accéder au cookie nom.

Supprimer un cookie

- Il n'existe pas une fonction dédiée à la suppression d'un cookie.

- Deux solutions peuvent être utilisées :

  - Renvoyer le cookie grâce à la fonction setcookie() en spécifiant uniquement le nom du cookie à supprimer.

  - Envoyer un cookie dont la date d'expiration est passée en spécifiant par exemple time()-1

Fonctionnalités de base de PHP

Les exceptions

- Une exception est une exception à la règle une erreur à gérer.
- En PHP la classe de base des exceptions est la classe **Exception**.
- Une exception peut être lancée **(throw)** et attrapée **(catch)** dans PHP.
- Le code devra être entouré d'un bloc **try** permettant de faciliter la saisie d'une exception potentielle.
- Chaque **try** doit avoir au moins un bloc **catch** ou **finally** correspondant.

Fonctionnalités de base de PHP

Les exceptions

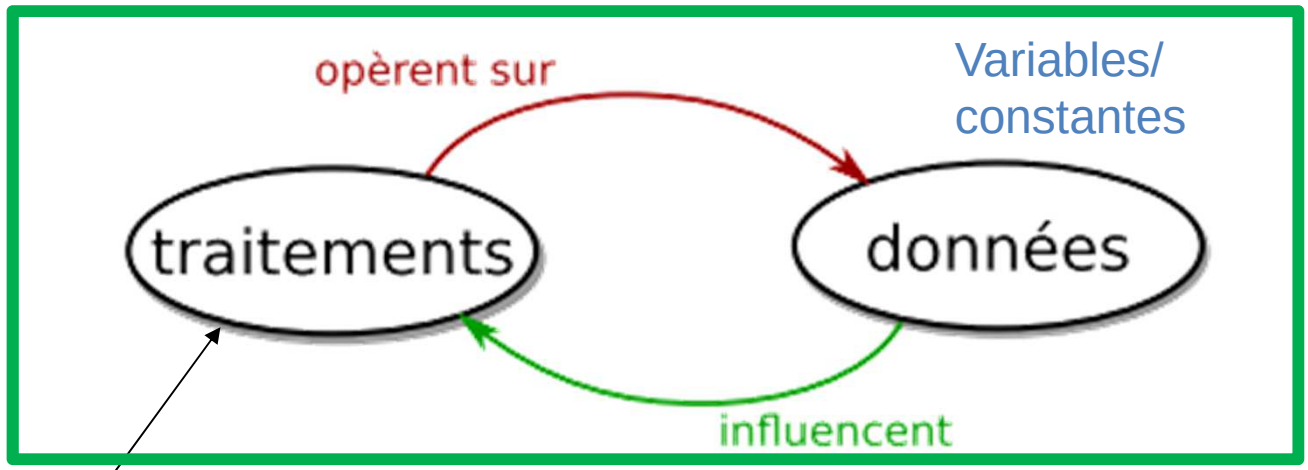
```php
<?php
// fonction qui retourne a/b
function quotient($a, $b) {
    if (!$b) {
        throw new Exception('Impossible de diviser par zéro.');
    }
    return $a/$b;
}
try {
    echo quotient(5,10) . "\n";
    echo quotient(3,0) . "\n";
} catch (Exception $e) {
    echo 'Exception catché : '.  $e->getMessage(). "<br>";
}
// Continue execution
echo "Je serais toujours la !<br>";
?>
```

POO PHP5

La programmation impérative/procédurale (rappel)

- Dans les scripts que vous avez écrits jusqu'à maintenant, nous avons vu les notions
  - de variables/types de données
  - et de traitement de ces données

Séparation entre le traitement et les données



ORIENTEE OBJETS

Fonctions ← arguments

Programmation orientée Objet

POO PHP5

Définition d'une classe et d'un objet

- Une classe en orientée objet représente un regroupement d'attributs (variables) et de méthodes (fonctions).

- Un objet est une instance de classe.

Classe

Objet

POO PHP5

Définition d'une classe et d'un objet

```
class NomClasse
{
  // Contenu de la classe
}
```

Déclaration d'un attribut

```
class NomClasse
{
  VisibilitéDeLaVariable $_nomVariable ;
  // Contenu de la classe
}
```

Exemple :

```
class Gateau
{
  private $_parfum;
}
```

POO PHP5

Définition d'une classe et d'un objet

Déclaration d'une méthode

```
class NomClasse
{
  VisibilitéDeLaVariable $_nomVariable ;
  VisibilitéDeLaFonction function nomDeLaFonction ;
}
```

Exemple :

```
class Gateau
{
  private $_parfum;
  public function appliquerGlacage(){}
  // Contenu de la classe
}
```

POO PHP5

Visibilité des attributs et des méthodes

- La visibilité permet de définir de quelle manière un attribut ou une méthode d'un objet est accessible.
- Les 3 niveaux de visibilité en PHP sont :
  - public (comportement par défaut) : Accessible partout
  - private : Accessible au sein de la classe elle même
  - Protected : Accessible au sein de la classe elle-même, ainsi qu'aux classes qui en héritent, et à ses classes parentes.
- Les objets de mêmes types ont accès aux membres privés et protégés les uns des autres.

POO PHP5

Créer un objet

- La création d'un objet à partir d'une classe est l'instanciation.
- L'instanciation se fait l'aide de new();
  - Exemple $monObjet = new MaClasse();
- New retourne l'identifiant de l'objet crée, donc $monObjet ne contient réellement l'objet mais son **identifiant**.
- Que fait alors $monNouvelObjet=$monObjet ?
- Afin de cloner un objet on utilise le mot clé **clone.**
  - $monNouvelObjet= clone $monObjet.
- Ceci permettra d'avoir **deux objets différents** avec les mêmes propriétés.
- Afin de parcourir l'ensemble des **propriétés visibles** d'un objet, nous pouvons utiliser le **foreach**.
  - Exemple :

```
foreach($this as $cle => $valeur) {
    print "$cle => $valeur\n";
}
```

Cet exemple permet à **chaque itération** d'avoir dans la variable $cle le nom de l'attribut et dans la variable $valeur sa valeur.

POO PHP5

- Les attributs et méthodes **statiques** sont des **attributs et méthodes de classe**. Elles sont **accessibles via la classe** sans avoir besoin d'instancier un objet.
- Elles sont déclarées avec le mot clé **static**.
- Pour accéder à un attribut ou méthode statique on écrit le nom de la classe suivi de l'opérateur de résolution de portée « **::** ».
- Un attribut statique est un attribut partagé par tout le monde. UNE SEULE COPIE.
- Interdit d'utiliser this !!!!!!!!!!!!!!!!!!!! qui représente la référence de l'objet.
- « L'équivalent » de this pour la classe est **self**.

POO PHP5

Attributs et méthodes statiques

```php
class MaStatic
{
    private $x;
    private static $static1=0;

    /**
     * MaStatic constructor.
     */
    public function __construct()
    {
        self::$static1++;
    }

    public static function nbInstance(){
        echo self::$static1;
    }
}
```

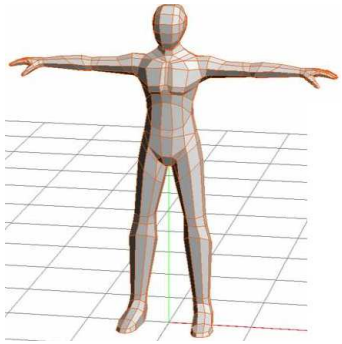
```php
$ma1= new MaStatic();
$ma2= new MaStatic();
$ma3= new MaStatic();

MaStatic::nbInstance();
```

POO PHP5

Les constantes de classe

- Les **constantes de classes** sont des attributs constants et qui appartiennent la classe et non à l'objet.
- Définit avec le mot clé **const**.
- Accessible via l'opérateur de résolution de portée « **::** ».
- Permettent d'éviter des **données muettes**.
- Exemple const PARIS=« Aéroport Charle de Gaulle»;

Commençons par un exemple concret : Les personnages de jeux vidéo



Héritage

Class Guerrier

string nom
int energie
int duree_vie

Arme arme

Rencontrer(Personnage)

Class Voleur

string nom
int energie
int duree_vie

Rencontrer(Personnage)

Voler(Personnage)

Class Magicien

string nom
int energie
int duree_vie

Baguette baguette

Rencontrer(Personnage)

Class Sorcier

string nom
int energie
int duree_vie

Baguette baguette

Baton baton

Rencontrer(Personnage)

PROBLEMES ?

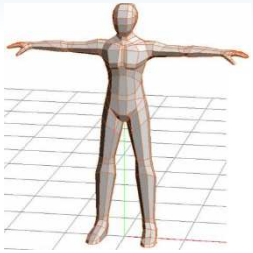
- ✓ Duplication de codes
- ✓ Problèmes de maintenance : Supposons qu'on veuille changer le nom ou le type d'un attribut, il faudra le faire pour chacune des classes !!!!!

Solution : Héritage

Héritage

Class Personnage

Pourquoi ne pas regrouper les caractéristiques en commun dans une super classe ?



string nom
int energie
int duree_vie

Rencontrer(Personnage&)

Class Voleur

Class Magicien

Class Guerrier

Voler(Personnage&)

Baguette baguette

Arme arme

Class Sorcier

Baton baton

Héritage

- En PHP pour dire qu'une classe A hérite d'une classe B on utilise le mot clé **extends**.
- La classe fille héritera de toutes les méthodes et les attributs de la classe mère mais elle ne pourra accéder qu'aux attributs public et protected.
- Une classe fille peut redéfinir les méthodes de la classe mère.
- Pour accéder à une méthode redéfinie de la classe mère on utiliser l'objet parent et l'opérateur de portée **parent::**.

POO PHP5

Héritage

```php
class Voiture
{
    private $matricule;
    private $nbChevaux;
    private $couleur;
    private $vitesse;
    public function
__construct($matricule, $nbChevaux,
$couleur, $vitesse)
    {
        $this->matricule = $matricule;
        $this->nbChevaux = $nbChevaux;
        $this->couleur = $couleur;
        $this->vitesse = $vitesse;
    }
public function tableauDeBord(){
    $date=new DateTime('NOW');
    echo'Bonjour Nous sommes le '.$date-
>format('Y-m-d').' <br>';
    echo'Mon matricule : '.$this-
>getMatricule().' <br>';
    echo'Vitesse actuelle : '.$this-
>getVitesse().' <br>';
}
    public function getMatricule()
    {
        return $this->matricule;
    }
public function setMatricule($matricule)
    {
        $this->matricule = $matricule;
    }
```

```php
public function getNbChevaux()
    {
        return $this->nbChevaux;
    }
public function
setNbChevaux($nbChevaux)
    {
        $this->nbChevaux = $nbChevaux;
    }
public function getCouleur()
    {
        return $this->couleur;
    }
public function setCouleur($couleur)
    {
        $this->couleur = $couleur;
    }
public function getVitesse()
    {
        return $this->vitesse;
    }
public function setVitesse($vitesse)
    {
        $this->vitesse = $vitesse;
    }
public function accelerer(){
     $this->vitesse+=10;
    }
    public function freiner(){
        $this->vitesse-=10;
    }
}
```

- Afin de forcer la classe à n'être qu'un modèle on la transforme en classe abstraite. Une classe abstraite est donc une classe non instanciable. Elle ne sert qu'a être une classe mère. On ne peut pas créer une voiture sans marque dans notre exemple.

- Une classe abstraite peut contenir des méthodes « normales » et des méthodes abstraites.

- Une méthode abstraite est une méthode non définie dans une classe abstraite. Elle sert à obliger toute classe héritant de la classe mère de redéfinir cette fonction.

- Une méthode finale est une méthode qui ne peut être redéfinie par une classe fille.

POO PHP5

Héritage

```php
Abstract class VoitureAbstraite
{
private $matricule;
private $nbChevaux;
private $couleur;
protected $vitesse;
public function __construct($matricule, $nbChevaux, $couleur,
$vitesse)
{
    $this->matricule = $matricule;
    $this->nbChevaux = $nbChevaux;
    $this->couleur = $couleur;
    $this->vitesse = $vitesse;
}
//Méthode Finale
final public function quatreFeu(){
    echo "Les quatres feux clignotent !!!";
}
//Méthode abstraite
Abstract public function reglageMoteur();
public function tableauDeBord(){
    $date=new DateTime('NOW');
    echo'Bonjour Nous sommes le '.$date->format('Y-m-d').' <br>';
    echo'Mon matricule : '.$this->getMatricule().' <br>';
    echo'Vitesse actuelle : '.$this->getVitesse().' <br>';
}
….
```

POO PHP5

Héritage

```php
class Megane2 extends VoitureAbstraite
{
    public function radarDeRecul()
    {
        echo "Attention Vous allez touchez un obstacle freinez s'il vous
plait";
    }
    // Fonction Statique si on la défini pas il y aura une erreur
    public function reglageMoteur()
    {
        echo " Je régle mon moteur";
    }

    public function tableauDeBord()
    {
        parent::tableauDeBord(); //
        echo "Je suis une Megane";
    }

    public function freiner()
    {
        parent::freiner();
        echo "je dispose de l'abs ";
        if ($this->getVitesse() > 100)
            $this->vitesse -= 30;
    }

    public function bluetooth()
    {
        echo 'bluetooth Activé votre téléphone est maintenat connecté';
    }
}
```

POO PHP5

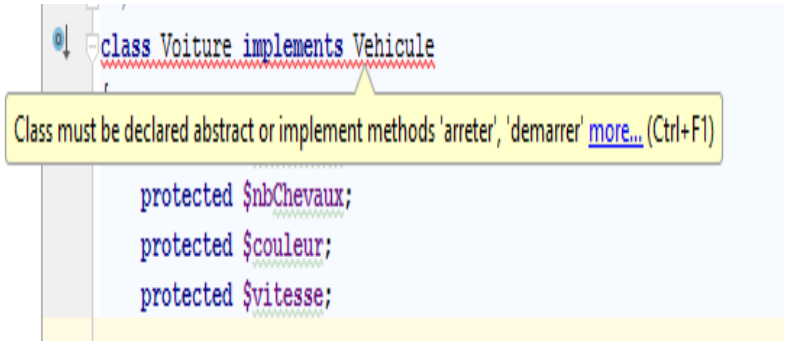
Interface

- Une **interface** est une classe qui permet de spécifier quelles méthodes les classes implémentant cette interface doivent implémenter.
- C'est un **contrat** permettant de préciser exactement le modèle à créer.
- Une interface est un **ensemble de prototype** de méthodes publiques.
- Une interface est déclarée comme une classe. Cependant la place du mot clé class vous devez mettre le mot clé **interface**.
- Afin de spécifier qu'une classe doit implémenter une interface on ajoute devant le nom de la classe le mot clé **implements** suivi du nom de l'interface.
- Une classe implémentant une interface **doit obligatoirement respecter la totalité du contrat**. **Toutes les méthodes** de l'interface doivent être **implémentées**.

POO PHP5

Interface

```php
interface Vehicule
{
    public function demarrer();
    public function arreter();
}
```




```php
class Voiture implements Vehicule
{
    protected $matricule;
    protected $nbChevaux;
    protected $couleur;
    protected $vitesse;

    public function demarrer()
    {
        // TODO: Implement demarrer() method.
    }

    public function arreter()
    {
        // TODO: Implement arreter() method.
    }
```

Conventions de nommage

- Nom de la classe DOIT être déclaré comme StudlyCaps. Le nom de la classe doit être descriptif. Il est conseillé d'éviter les abréviation tel que Pers pour parler d'une personne.
- Pour les noms composées utiliser le séparateur « _ » et les majuscules pour passer d'un nom à un autre ou uniquement les majuscules selon la norme que vous suivez.

- Les méthodes et les attributs doivent être nommées en utilisant le camelCase, $parfumGlacage, $getParfum(), ceci s'applique pour la visibilité non privées. Par convention les attributs avec une visibilité privée doivent avoir un nom commençant par « _ ». Exemple : $_parfum.

- Les constantes doivent avoir un nom qui est en totalité en Majuscule avec «_» en séparateur.

POO PHP5

Auto Chargement

- Il faut inclure une Classe afin de l'utiliser un script PHP en utilisant le mot clé **require.**
  - **Exemple : require ConnexionBD.php**
- Afin d'éviter de charger manuellement chaque classe à utiliser dans un script, il vaut mieux utiliser le concept d'**autoload**.
- PHP possède une pile d'autoloads qui permet de lister les fonctions à utiliser pour gérer le chargement automatique des classes instanciées et non déclarées.
- La solution est donc d'ajouter une fonction qui permet de charger ces classes.
- Afin d'ajouter une fonction à la Pile de fonctions d'auto chargement on utiliser la fonction **spl_autoload_register( )** qui prend en paramètre une chaine de caractère représentant le nom de la fonction.

POO PHP5

Auto Chargement

```php
function loadClass($maClasse)
{
    require $maClasse . '.php';
}
```

PHP Accès à la BD

Extensions gestion BD

- Pour la gestion des Bases de données, PHP offre plusieurs extensions :
  - L'extension **mysql_** : Ensemble de fonction commençons par mysql_ et permettant de communiquer avec mysql. Elle sont devenue obsolète.
  - L'extension **mysqli_** : ce sont des fonctions améliorées d'accès à MySQL.
  - L'extension **PDO** : Outil complet qui permettant une abstraction du type de la base de données traitée permettant ainsi de se connecter aussi bien à MySQL que PostgreSQL ou Oracle.

PHP Accès à la BD

Tableau comparatif

Tableau comparatif

Src : http://www.php.net/manual/fr/mysqli.overview.php

	Extension MySQL	Extension mysqli	PDO (avec le pilote PDO MySQL Driver et MySQL Native Driver)
Version d'introduction en PHP	Avant 3.0	5.0	5.0
Inclus en PHP 5.x	Oui, mais désactivé	Oui	Oui
Statut de développement MySQL	Maintenance uniquement	Développement actif	Développement actif depuis PHP 5.3
Recommandée pour les nouveaux projets MySQL	Non	Oui	Oui
L'API supporte les jeux de caractères	Non	Oui	Oui
L'API supporte les commandes préparées	Non	Oui	Oui
L'API supporte les commandes préparées côté client	Non	Non	Oui
L'API supporte les procédures stockées	Non	Oui	Oui
L'API supporte les commandes multiples	Non	Oui	La plupart
L'API supporte toutes les fonctionnalités MySQL 4.1 et plus récent	Non	Oui	La plupart

Nous nous focalisons donc sur **PDO**

PHP Accès à la BD

PDO

- PDO : **P**HP **D**ata **O**bjects
- Extension fournissant des services d'accès aux bases de données.
- Fournie avec plusieurs drivers (MySQL, sqlite, PostgreSQL
- Disponible par défaut à partir des serveurs PHP 5.1.0

PHP Accès à la BD

Connexion à une BD

➢ Pour se connecter à une base de données on instancie un objet PDO de la façon suivante :

```
$maDb_connexion = new
PDO('mysql:host=localhost;dbName=nomDeLaBase', 'userName',
'motDePasse');
```

On crée donc une nouvelle instance de PDO qu'on récupère dans la variable $maDb_connexion. Le constructeur nécessite trois paramètres :

- le driver utilisé, l'adresse du serveur et le nom de la base noté ainsi driver:hot=serveur; dbName=nomDeLaBase'
- le nom d'utilisateur à utiliser pour la connexion au serveur
- le mot de passe du dit utilisateur

Ces informations différent un peu selon le pilote.

PHP Accès à la BD

Connexion à une BD

- Afin de gérer les erreurs liées à la Connexion à la BD, il faut capturer les erreurs de type PDOException.

```
try {
    $db_connexion = new
PDO('mysql:host=localhost;dbname=user1', 'user1',
'motdepass');
}
catch (PDOException $e)
{
    print "Erreur : " . $e->getMessage();
    die();
}
```

Pattern Design singleton

- But : Singleton garantit qu'une classe n'a qu'une seule instance et fournit un point d'accès global à cette instance.
- Principe
  - Empêcher les développeur d'utiliser le ou les constructeurs de la classe : déclarer privé tous les constructeurs de la classe.
  - Problème : la classe n'est plus instanciable que par elle-même.
  - Solution : Construire un pseudo constructeur à travers une méthode static. Par convention il sera appelé getInstance.
  - On crée un attribut statique stockant l'unique instance de la classe.
  - Dans getInstance on teste si cet attribut est nul
    - Si null on instancie un objet et on le retourne
    - Sinon on retourne l'instance existante

PHP Accès à la BD

Connexion à une BD : Pattern Design singleton

```php
<?php
class ConnexionBD
{
private static $_dbname = "bdphp5";
private static $_user = "root";
private static $_pwd = "";
private static $_host = "localhost";
private static $_bdd = null;
private function __construct()
{
 try {
        self::$_bdd = new PDO("mysql:host=" . self::$_host . ";dbname="
.self::$_dbname . ";charset=utf8", self::$_user, self::$_pwd,
array(PDO::MYSQL_ATTR_INIT_COMMAND => 'SET NAMES UTF8'));
    } catch (PDOException $e) {
        die('Erreur : ' . $e->getMessage());
    }
}
 public static function getInstance()
    {
        if (!self::$_bdd){
            new ConnexionBD();
            return (self::$_bdd);
    }return (self::$_bdd);
    }
}
```

PHP Accès à la BD

Connexion à une BD : afficher les erreurs

- Afin d'afficher les détails des erreurs liées à la BD, il faut activer le suivi de ces erreurs lors de la connexion à la BD.
- Exemple :

```
try {
    $db_connexion = new
PDO('mysql:host=localhost;dbname=user1', 'user1', 'motdepass',
array(PDO::ATTR_ERRMODE =>
PDO::ERRMODE_EXCEPTION)
);
}
catch (PDOException $e)
{
    print "Erreur : " . $e->getMessage();
    die();
}
```

PHP Interroger une BD

Requête simple

- Afin d'interroger une BD via PDO, nous utilisons la méthode query qui prend en paramètre la requête à exécuter.
- Exemple :

```
$req=« select * From maTable »;
$reponse = $_bdd->query($req);
```

- La variable $reponse contiendra un objet contenant la réponse de MySQL qui n'est pas directement exploitable.
- Pour exploiter ces données nous utilisons la méthode fetch qui retourne une ligne ou fetchAll qui retourne un tableau contenant toutes les lignes du jeu d'enregistrements
- L'un des paramètres des méthodes fetch et fetchAll est l'attribut fetch_style qui permet de spécifier le type de la valeur de retour de fetch et fetchAll

PHP Interroger une BD

Fetch style

- Contrôle comment la prochaine ligne sera retournée à l'appelant. Cette valeur doit être une des constantes PDO::FETCH_*.
- PDO::FETCH_BOTH (défaut): retourne un tableau indexé par les noms de colonnes et aussi par les numéros de colonnes, commençant à l'index 0, comme retournés dans le jeu de résultats
- PDO::FETCH_ASSOC : retourne un tableau indexé par le nom de la colonne comme retourné dans le jeu de résultats
- PDO::FETCH_NUM : retourne un tableau indexé par le numéro de la colonne comme elle est retourné dans votre jeu de résultat, commençant à 0
- PDO::FETCH_OBJ : retourne un objet anonyme avec les noms de propriétés qui correspondent aux noms des colonnes retournés dans le jeu de résultats
- …

PHP Interroger une BD

Fetch style

- Exemple de parcours en utilisant des objets :

```
$games = $rep->fetchAll(PDO::FETCH_OBJ);

foreach($games as $game) :
    echo $game->nom." - ".$game->commentaires."<br>";
endforeach;

$rep->closeCursor();
```

PHP Interroger une BD

Requête paramétrable

- Afin de paramétrer une requête nous pouvons utiliser deux méthodes:
  - Paramétrage manuelle en concaténons les paramètres dans la requête. ENORME FAILLE DE SECURITE SQL INJECTION
  - les requêtes préparées
- Deux types de requêtes préparées :
  - En utilisant les marqueurs « ? »
  - En utilisant les marqueurs nominatifs

PHP Accès à la BD

Requête préparée

- Pour simplifier les choses une requête préparé est une requête dont les paramètres sont insérés dans la fonction lors de l'exécution.

- Elle est effectuée en 2 étapes :

  - Préparer la requête à l'aide de la méthode prepare

  - Transmettre les paramètres dans un tableau et exécuter la requête préparée à l'aide de la méthode execute

  - Les paramètres sont indiqués dans l'ordre d'apparition dans la requête préparée

  - Le contenu des variables est automatiquement sécurisé pour prévenir les risques d'injection SQL.

Exemple :

```
$req = $bdd->prepare('SELECT * FROM personne WHERE nom = ?  AND age <= ? ORDER BY cin');
$req->execute(array($nom, $age));
```

PHP Accès à la BD

Requête préparée

- Pour simplifier les choses une requête préparé est une requête dont les paramètres sont insérés dans la fonction lors de l'exécution.

- Elle est effectuée en 2 étapes :
  - Préparer la requête à l'aide de la méthode prepare
  - Transmettre les paramètres dans un tableau et exécuter la requête préparée à l'aide de la méthode execute
  - Les paramètres sont indiqués dans l'ordre d'apparition dans la requête préparée
  - Le contenu des variables est automatiquement sécurisé pour prévenir les risques d'injection SQL.

Exemple :

```
$req = $bdd->prepare('SELECT * FROM personne WHERE nom = ?  AND age <= ? ORDER BY cin');
$nom=« test »;$age=« 10 »;
$req->execute(array($nom, $age));
```

PHP Accès à la BD

Requête paramétrable

- Requête paramétrable avec des marqueurs nominatifs
- Afin de rendre la requête préparée plus lisible, ion peut remplacer les ? Par des marqueurs nommés
- Un marqueur nommé est un nom précédé par « : »
- Exemple :

```
$req = $bdd->prepare('SELECT * FROM personne WHERE nom = :nom  AND age <= :age ORDER BY cin');
$req->execute(array('nom'=>$nom, age=>$age));
```

L'ordre des paramètres n'a plus d'importance vu que nous utilisons des tableaux associatifs.

PHP Accès à la BD

Requête paramétrable

- On peut aussi utiliser la méthode bindValue qui prend en paramètres : 1 - rang de l'attribut si on n'utilise pas d'attribut nominatif sinon son nom
  2 - Le contenu
  3 – Le type (PARAM_STR, PARAM_BOOL, PARAM_INT)

  - **PDO::PARAM_BOOL** Représente le type de données booléen.
  - **PDO::PARAM_NULL** Représente le type de données NULL SQL.
  - **PDO::PARAM_INT** Représente le type de données INTEGER SQL.
  - **PDO::PARAM_STR** Représente les types de données CHAR, VARCHAR ou les autres types de données sous forme de chaîne de caractères SQL.
- Exemple :

```
$req = $bdd->prepare('SELECT * FROM personne WHERE nom = ? AND age <= ? ORDER BY cin');
$req->bindValue(1,"Aymen",PDO::PARAM_STR);
$req->bindValue(2,20,PDO::PARAM_INT);
$req->execute();
```

PHP Accès à la BD

Requête paramétrable

- Afin de récupérer le nombre d'enregistrement retourné par la requête on utilise la méthode rowCount.

Exemple :

```
$req="select * maTable";
$rep = $bdd->query($req);
echo "le nombre d'enregistrements est :".$rep->rowCount();
```

- Afin de récupérer l'id du dernier enregistrement, on utilise la méthode lastInsertedId.

Remarque : Ca ne marche qu'après un INSERT.

Exemple :

```
echo "le dernier id est :".$bdd->lastInsertId()."<br>";
```

PHP Accès à la BD

Ajout d'enregistrement BD

- Afin d'ajouter, modifier et supprimer un enregistrement dans la Base de données PDO nous offre la méthode **exec**.
- Cette méthode prend en paramètre la requête à exécuter.
- On peut utiliser la méthode prépare afin de préparer la requête à exécuter.
  - $req= $bdd->**prepare**(« La requête à préparer »)
- Une fois la requête préparée, on utilise la méthode **execute** en lui passant un tableau associatif contenant la liste des paramètres.

```
$req= $bdd->prepare("insert into matable
(`champ1`, `champ2`, `champn`) VALUES
(:val1,:val2,:valn)");

$req->execute(array(
    'val1'=>'val1',
    'val2'=>'val2',
    'valn'=>5
));
```

PHP Accès à la BD

Modification d'enregistrement

- Même fonctionnement que l'ajout.
- Requête update.

```
$req= $bdd->prepare(« update matable set
champ1=:val1, champ2= :val2, champ3=
:champ3 where champ_condition= :cnd");

$req->execute(array(
    'val1'=>'newval1',
    'val2'=>'newval2',
    'valn'=>7,
    'cnd'=>'valCnd',
));
```

PHP Accès à la BD

Suppression d'enregistrement

- Même fonctionnement que la suppression
- Requête delete.

```
$req= $bdd->prepare(« delete from matable
where champ_condition= :cnd");

$req->execute(array(
    'cnd'=>'valCnd',
));
```

PHP

AYMEN SELLAOUTI
aymen.sellaouti@gmail.com