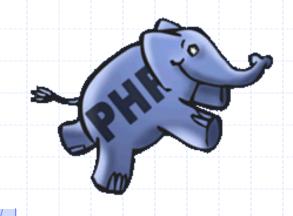
## Cours de PHP



Jean-Luc Collinet

Haute École Léonard de Vinci : Paul Lambin

Bloc 1 du Bac en Informatique



#### Bases de données

- Essentielles pour stocker et gérer les informations
- Exemple du CMS
  - Content Management System
    - Joomla, Drupal
    - Prestashop
- Exemple du CRM
  - Customer Relationship Management
    - SuiteCRM, Vtiger, Odoo CRM

Diapositive 2 BINV-1050 Cours 3



#### **SGBD**

- Système de Gestion de Base de Données
  - SQLite
    - Moteur embarqué (extension de PHP)
  - MySQL
  - MariaDB
    - Il s'agit d'un fork communautaire de MySQL
  - Et bien d'autres...
    - PostgreSQL

Diapositive 3



Cours 3

#### PDO

- PHP Data Objects
  - http://fr.php.net/manual/fr/intro.pdo.php
- Couche d'abstraction du moteur SGBD
  - Orienté objet
  - Séparer la couche des traitements de la couche base de données proprement dite
  - Migrer vers un autre SGDB est ainsi facilité
- Driver à installer selon le SGBD employé
  - http://fr.php.net/manual/fr/pdo.drivers.php

Diapositive 4 BINV-1050



# Opérations courantes sur une Db

- Ouvrir une connexion
- Faire une requête SQL
  - CREATE
  - SELECT, INSERT, UPDATE, DELETE
- Traiter le résultat de la requête
- Fermer la connexion

Diapositive 5 BINV-1050 Cours 3



# Développer une classe Db

```
class Db {
    private $_db;
```

- # Cette classe va contenir toutes les
- # méthodes pour faire des requêtes
- # sur la base de données
- # \$\_db représente l'objet « connexion
- # à la base de données »

Diapositive 6

BINV-1050



## Comment créer une connexion ?

```
# Connexion à la base de données
# par instanciation de la classe PDO
$this->_db = new
  PDO('mysql:host=localhost;dbname=bdbn
  ;charset=utf8', 'root', ");
                # Le serveur MySQL : ici localhost
        # Le nom de la base de données : ici bdbn
                   # Le nom d'utilisateur : ici root
```

Diapositive 7 BINV-1050 Cours 3

# Le mot de passe : ici pas de mot de passe, root sur



#### Le constructeur de la classe Db

```
private function __construct() {
  try {
     $this->_db = new
     PDO('mysql:host=localhost;dbname=bdbn;charset=utf8',
'root',");
     # Ne jamais écrire d'espace dans une string de connexion
     $this->_db->setAttribute(PDO::ATTR_ERRMODE,
     PDO::ERRMODE_EXCEPTION);
  catch (PDOException $e) {
     die ('Erreur de connexion à la base de données : '.
           $e->getMessage());
```

Diapositive 8

BINV-1050

Cours 3



# Attributs de l'objet PDO

Diapositive 9

BINV-1050



## Pattern Singleton

```
class Db {
   private static $instance = null;
   private $_db;
   public static function getInstance() {
      if (is_null(self::$instance)) {
          self::$instance = new Db();
   return self::$instance;
```

Diapositive 10

BINV-1050



#### Besoin de la connexion Db?

- Écrire tout simplement
   \$db = Db::getInstance()
   dans le contrôleur maître index.php
- Passer ensuite \$db en paramètre à un sous-contrôleur ayant besoin d'une connexion à la base de données \$controller = new AmiController(\$db);

Diapositive 11 BINV-1050 Cours 3



#### Besoin de la connexion Db?

- Dans le sous-contrôleur, un attribut private \$\_db;
- Un appel à une fonction de la classe Db \$tabamis=\$this->\_db->select\_amis();

Diapositive 12 BINV-1050 Cours 3



# Faire une requête SELECT

```
Dans une méthode de la classe Db,
Écrire $query = 'SELECT no, nom FROM ami';
$ps = $this->_db->prepare($query);
$ps->execute();
```

- # À privilégier pour la sécurité : la fonction prepare suivie de la fonction execute
- # \$ps est un objet de la classe PDOStatement

http://php.net/manual/fr/class.pdostatement.php

Représente une requête préparée et, une fois exécutée, le jeu de résultats associé.

Diapositive 13 BINV-1050 Cours 3



## Y a-t-il des résultats?

```
if ($ps->rowcount()!=0) {
    # Il y a au moins une ligne
    # dans les résultats
    # Parcourir les résultats dans $ps
    # avec un foreach ou un while...
    # et la fonction fetch... (cf. dia suivante)
```

Diapositive 14 BINV-1050 Cours 3



## Parcourir les résultats

- Et retourner un tableau d'objets
   while (\$row = \$ps->fetch()) {
   \$tableau[] =
   new Ami(\$row->no, \$row->nom);
   }
- Attention, initialiser le tableau !\$tableau = array();
- Bien sûr, écrire une classe Ami!

Diapositive 15 BINV-1050 Cours 3



# La fonction var\_dump()

- Utilisez var\_dump(\$unevariable);
- « Dump » d'une variable
- Ecrit dans la source HTML à l'endroit de l'appel de la fonction le type et la valeur de la variable passé en paramètre
- Peut être employée pour débugger un contrôleur ou une méthode de la classe Db

Diapositive 16 BINV-1050 Cours 3



# Faire une requête INSERT

```
$query = 'INSERT INTO ami (nom)
         VALUES ('.
             $this->_db->quote($nom)
          . ')';
$this->_db->prepare($query)->execute();
      # La méthode quote permet de gérer
```

# les quotes pour \$nom paramètre de la

# méthode d'insertion et valeur insérée

Diapositive 17 BINV-1050



## Faire un INSERT avec bindValue

# \$qp est un objet de la classe PDOStatement

# \$nom est un paramètre de la méthode

# d'insertion et ainsi la valeur insérée

Diapositive 18 BINV-1050 Cours 3



# Attaques par injection SQL

- Possibilité d'entrer des valeurs « de programmation » dans un formulaire qui peuvent attaquer la base de données...
- Appeler un « PDO::prepare() » suivi d'un « PDOStatement::execute() » aiderait à prévenir les attaques par injection SQL
- Google it!

Diapositive 19 BINV-1050 Cours 3



## Sécuriser la faille XSS

- Db::getInstance()->insert\_ami(\$\_POST['nom']);
- JavaScript à introduire dans un formulaire pour voir s'il y a une faille XSS :
  - <script>
     alert('Il y a une faille XSS')</script>
- Nous allons sécurisez la faille XSS lors de l'affichage des données avec la fonction htmlspecialchars écrite dans le modèle...

Diapositive 20 BINV-1050 Cours 3



# Sécurisation dans le modèle, en vue de l'affichage

- Dans la classe Ami,
   écrivons des getteurs spécifiques
   qui seront utilisés pour éviter la faille XSS
- public function html\_nom() {
   return htmlspecialchars(\$this->\_nom);
  }
- La fonction va remplacer < par &lt; et > par > dans le code source HTML

Diapositive 21 BINV-1050 Semaine 3



#### Faire un UPDATE

\$this->\_db->prepare(\$query)->execute();

Diapositive 22 BINV-1050 Cours 3



## Comment fermer la connexion ?

C'est automatique à la fin du script PHP

Diapositive 23 BINV-1050 Cours 3



# PDOException à titre informatif

```
try {
  $query = ' ... ';
  $dbh->prepare($query)->execute();
catch (PDOException $e) {
     $msgerreur = $e->getMessage();
```

Diapositive 24

BINV-1050

Cours 3



#### **Architecture Db**

- Db proprement dite en MySQL, accessible via phpMyAdmin
- Une seule classe Db regroupe toutes les requêtes aux données
  - models/Db.class.php
- Appliquer le Pattern Singleton pour la connexion à la Db
- Une classe par table (ex. la classe Ami)

Diapositive 25 BINV-1050 Cours 3



## Exercices: gestion des livres

- Création de la Db et de la table des livres
  - En mode « PHP système » (non MVC)
  - A l'aide de phpMyAdmin (logiciel inclus avec Wampserver)
- Insertions à l'aide de phpMyAdmin
- Classe Db, Pattern Singleton, classe Livre
- Affichage de la table des livres (SELECT)
- Ajout d'un livre via formulaire (INSERT)
- Recherche de livre(s) selon un mot clé

Diapositive 26 BINV-1050 Cours 3



## Pour bien gérer les lettres accentuées

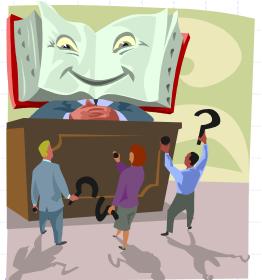
- Que la base de données, les tables et les champs soient bien tous en UTF8 : interclassement utf8\_general\_ci
- ;charset=utf8 termine la string de connexion à la Db

Diapositive 27 BINV-1050 Semaine 3



## Bonnes expérimentations à tous

- Il y a ceux qui voient la réalité et qui disent : Pourquoi ?
- Et il y a ceux qui rêvent de l'impossible et qui disent : Pourquoi pas ?



George Bernard Shaw

Diapositive 28 BINV-1050 Cours 3