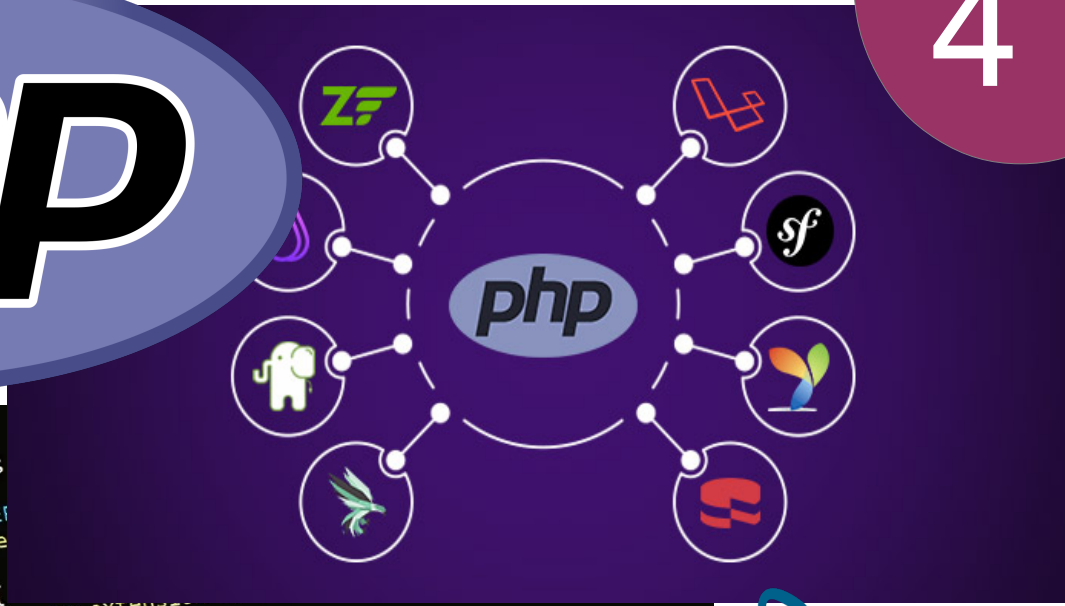
php

4




```
 * @var boolean
 */
define('PSI_INTERNAL_XML', false);

if (version_compare("5.2", PHP_VE
    die("PHP 5.2 or greater is re

}
if (!extension_loaded("pcre")) {
    die("phpSysInfo requires the pcre extens
        properly.");

}

require_once APP_ROOT.'/includes/autoloader.inc.php';

// Load configuration
require_once APP_ROOT.'/config.php';

if (!defined('PSI_CONFIG_FILE') || !defined('PSI_DEBUG')) {
    $tpl = new Template("/templates/html/error_config.html");
    echo $tpl->fetch();
    die();
```

MySQL®

Messmer Lucie

Dans ce cours :

- Découverte du SGBD
- MySQL
- Créattion de la base de données
- Création des tables
- Les différents types
- Requêtes simples

Exercice PHP3

Système de gestion de base de données (SGBD)

Le **SGBD** est le logiciel permettant de **manipuler les données d'une base**. C'est grâce à lui qu'on va commander les interactions avec notre base de données pour y récupérer, ajouter, modifier, ou supprimer des données.

SQL

Le Structured Query Language (langage de requête structuré) est un langage informatique qui nous permet d'intéragir avec nos bases de données.

MYSQL

Ce SGBD est le plus connu et le plus utilisé. C'est avec celui-ci que nous allons travailler.

Pour utiliser une database déjà crée (un fichier .sql avec le details de la databse : création de la db, et des différentes tables), on utilise la commande suivante dans un terminal MySQL :

nom_de_la_base_de_donnees < nom_du_fichier.sql

ou alors :

use nom_base_de_donnees;

source fichier.sql;

L'utilisation du langage MySQL est très simple car très littérale.

Une instruction par ligne, chaque instruction finissant par **;**

Création de la base de données (BDD)

La base de données contiendra tous nos objets (utilisateurs, clients, objets mis en vente, etc).

Pour activer MySQL dans son terminal :

→ Ouvrir le terminal sur VSCode

→ éxécuter la commande **: mysql -u root -p**

Cette commande demandera à ce qu'on inscrive notre mot de passe + touche Enter. (Si aucun mot de passe, juste appuyer sur Enter).

Le terminal va changer légèrement, et à présent, toutes les commandes écritent seront du SQL (à finir par ;).

Pour créer sa database : **CREATE DATABASE nomdeladb ;**

Pour vérifier la liste des databases existantes : **SHOW DATABASES ;**

Pour utiliser la database nouvellement créé : USE nomdeladb ;

Créer une table

Une table est un tableau regroupant des informations liés à un objet. Imagineons un blog, dans ce blog, nous pourrions avoir la table utilisateur, article, etc.

Créer une table, c'est la définir, et donc définir ce qu'elle doit contenir.

On créé une table avec **CREATE TABLE nomdelatable ( ) ;**

exemple :

```
CREATE TABLE utilisateur (
id,
nom,
prenom,
email
);
```

Type de données

Lorsqu'on créé notre table et qu'on défini les éléments les constituants, il faut définir leur type, par exemple pour la table utilisateur :

nom du champ	type et options	description
id	PRIMARY KEY (option)	Champ spécial obligatoire dans toutes les tables. Indique à MySQL que ce champ sera l'identifiant permettant d'identifier les objets.
	INTEGER (type)	Champ numérique sous forme de nombre entier.
	NOT NULL (option)	Ce champ ne peut pas être nul.
	AUTO_INCREMENT (option)	Ce champ sera créé par MySQL automatiquement, pas besoin de s'en soucier ! MySQL va utiliser l'id précédent et y ajouter +1 lors de l'ajout d'un nouvel objet.
nom	VARCHAR(100) (type)	Champ sous forme de texte, limité à 100 caractères.
prenom	VARCHAR(100) (type)	Champ sous forme de texte, limité à 100 caractères.
email	VARCHAR(255) (type)	Champ sous forme de texte, limité à 255 caractères.
	NOT NULL (option)	Ce champ ne peut pas être nul.
	UNIQUE (option)	Ce champ ne peut pas avoir la même valeur en double.

exemple :

CREATE TABLE utilisateur (

id INTEGER NOT NULL AUTO_INCREMENT PRIMARY KEY,

nom VARCHAR(100),

prenom VARCHAR(100),

email VARCHAR(255) NOT NULL UNIQUE

);

Les types de données principales sont :

VARCHAR, TINYINT, SMALLINT, MEDIUMINT, INT, BIGINT, FLOAT, DOUBLE, DATE, REAL, DECIMAL, NUMERIC

On peut utiliser à nouveau la commande SHOW pour vérifier nos tables créer : **SHOW tables** ;

Pour avoir le détails d'une table : **DESCRIBE nomdelatable** ;

Connecter son application à sa DB

Syntaxe :

```
$db = new PDO("mysql:host=".DB_HOST.";dbname=".DB_NAME, DB_USERNAME, DB_PASSWORD);
```

On enregistre dans une variable notre connexion à notre database.
Cette connexion comprend notre serveur, le nom de la database, le nom d'utilisateur, et le mot de passe associé à cet utilisateur.

Dès que nous souhaiterons utiliser notre db, il faudra faire appelle à cette variable.

Il est fortement conseille de tester la connection a notre DB avant d'aller plus loin.

```
if ($db) {
    echo "Connexion réussi";
} else {
    echo "La connexion n'a pas aboutie";
}
```

Requêtes SQL principales

INSERT INTO

Cette requête permet d'insérer des éléments dans la database.

Syntaxe :

INSERT INTO nomdelatable ('nom', 'prenom', 'age') VALUES ('Dupond', 'Jean', 37) ;

On peut également insérer plusieurs éléments d'un coup à un table :

```
INSERT INTO `utilisateur` (`nom`, `prenom`, `age`)
VALUES
('Doe', 'John', 26),
('Smith', 'Jane', 76),
('Dupont', 'Sebastien', 52),
('Martin', 'Emilie', 49);
```

Backticks ` `	nom de table ou colonne
Guillemets	valeurs de type TEXT ou VARCHAR
Pas de guillemets	valeurs de type BOOLEAN, INTEGER, FLOAT

PHP

SELECT

Cette requête permet de récuperer des éléments de la database.

Syntaxe :

SELECT * FROM nomdelatable ; → on cherche à récupérer **tous** les éléments de la table nomdelatable.

On peut remplacer * (tout) par le nom des colonnes à récupérer

UPDATE

Cette requête permet de modifier des éléments de la database.

Syntaxe :

UPDATE nomdelatable SET unecolonne = "la modification qu'on souhaite apporter" WHERE `id` = '1' ;

Ici, on va modifier la colonne 'unecolonne' dans la table "nomdelatable" où l'id est égal à 1.

DELETE

Cette requête permet de supprimer des éléments de la database.

Syntaxe :

DELET FROM nomdelatable WHERE ` id `= '2' ; → on supprime un élément de la table nomdelatable où l'id = 2.

Envoyer des requêtes simples

Une fonction permet d'envoyer une requete SQL au SGBD et de récupérer un résultat :

- Pour les SELECT, ce sera un résultat brut sous forme de ressource.
- Pour les INSERT, UPDATE et DELETE, ce sera 'true' en cas de réussite de l'opération, 'false' dans le cas contraire.

MYSQL_QUERY

Syntaxe :

```
$sql = 'SELECT * FROM table';
// on a stocké ici la requête dans une variable nommée $sql
// il s'agit de code SQL stocké dans une variable PHP.
// PHP dispose d'une fonction pour exécuter ce code SQL :
$resultat = mysql_query($sql) or die('Erreur '.mysql_error());
// on envoie la requete (stockée dans $sql) grâce à la fonction
// mysql_query de PHP. Le résultat (les valeurs récupérées) sont
// stockées dans la variable $resultat.
while($data = mysql_fetch_array($resultat)) {
  echo ( $data['nom'].$data['prenom'].' ('.$data['statut'].')';
  echo ' date de naissance : '.$data['date'].;
}
// Une fois le résultat récupéré dans la variable (ici $resultat)
// il ne reste qu'à l'exploiter avec PHP. Le résultat étant (logiquement)
// stocké sous forme de tableau.
```