Cours PHP / Object BDD

Dans ce cours vous allez trouver tout le nécessaire pour se connecter à une base de donnée via PHP et exécuter vos premières requêtes ainsi que récupérer les résultats de cette dernière pour les afficher en PHP/HTML

1. L’Object PDO Php

Pour faciliter l’interaction entre PHP et Une Base de donnée les développeurs on créer une classe très pratique qui porte le nom de PDO. Cette classe en langage orienté objet nous permet de créer un objet de type PDO . Dans cette classe nous trouvons une partie Propriété très complète et une partie Méthode ( voir cours langage Objet ). Dans ce cours nous allons voir qu’une partie infime des propriétés et méthodes offerte par la classe PDO. Pour plus d’information => rdv dans la doc de la classe à cette adresse :

<http://php.net/manual/fr/class.pdo.php>

1. La création de l’objet PDO

Pour instancier un objet PDO ( le créer ) on utilise le mot clé "new" en php

$maBase=new PDO('mysql:host=localhost; dbname=test; charset=utf8','root', '');

$maBase est le nom de la variable ou sera stocké en mémoire l’objet de type PDO que vous venez de créer.

Mysql : c’est le nom du gestionnaire de base de donnée utilisé

Host = C’est l’adresse IP de la BDD

Dbname = c’est le nom de la base de votre BDD

Charset = c’est le type d’encodage caractère de votre BDD

Les deux derniers caractères sont le nom d’utilisateur et le mdp

3) Exécuter une requète sur la BDD

Pour exécuter une requête on doit utiliser l’objet PDO qui dans notre cas est stocké dans la variable $maBase . On utile donc la méthode adéquate qui est ici la méthode "query"

$LesDonneesBrutesDeLaBdd = $maBase->query("SELECT UPDATE DELETE Ect FROM maTable WHERE condition")

Pour accéder à la méthode (c’est une fonction) d’un objet on utilise le racourcie "->" et le nom de la méthode ( ici query(le requete) ).

La variable $LesDonneesBrutesDeLaBdd récupéra en brute les données reçu de la BDD

1. Travailler sur les résultats d’une requète de la BDD

PHP n’est pas capable d’afficher $LesDonneesBrutesDeLaBdd car il ne connaît pas au moment de la compilation les données qui seront envoyées en requête et ce n’est pas son langage. C’est donc au développeur d’anticiper le code pour afficher les résultats grâce toujours à l’objet PDO.

Pour cela on utilise la méthode fetch() ; qui va lire la première ligne des données brutes de la Base est en faire un tableau. Chaque fois qu’on utilise ma méthode fetch() celle ci va lire la ligne suivante des données brutes. C’est pour çà que l’on parcourt une boucle pour afficher les données brutes tour à tour.

$TableauDeDonnée = $LesDonneesBrutesDeLaBdd->fetch() ;

Ainsi $TableauDeDonnée est un tableau ou sont stockées les données d’une ligne en cours de traitement de $LesDonneesBrutesDeLaBdd

Pour finir , pour afficher la donnée on utilise le bon indice du tableau qui porte généralement un N° de colonne ou le nom du champs de la BDD

Exemple : echo $TableauDeDonnée["nom"] ; affichera le nom de la ligne en cours.

1. Algo rapide pour afficher toutes les données ligne par ligne.

while ($TableauDeDonnée = $ LesDonneesBrutesDeLaBdd ->fetch())

{

echo $TableauDeDonnée["nom"] ;

}

$LesDonneesBrutesDeLaBdd ->closeCursor();

1. gérer les erreurs de BDD

Pour exécuter un bout de code en php qui attrapera à coup sur une erreur on utilise la technique du Try Catch ( Try pour tester , Catch pour gérer les éventuelles erreurs )

Try{

//execution du code sur la BDD exemple

$maBase=new PDO('mysql:host=localhost; dbname=test; charset=utf8','root', '');

}

catch (Exception $erreur){

die('Erreur : '.$erreur ->getMessage());

}

Si le code n’a pas d’erreur on reste dans les { } du Try sinon on passe dans le catch et on gère l’erreur. Par exemple on peut l’afficher comme dans l’exemple ci-dessus