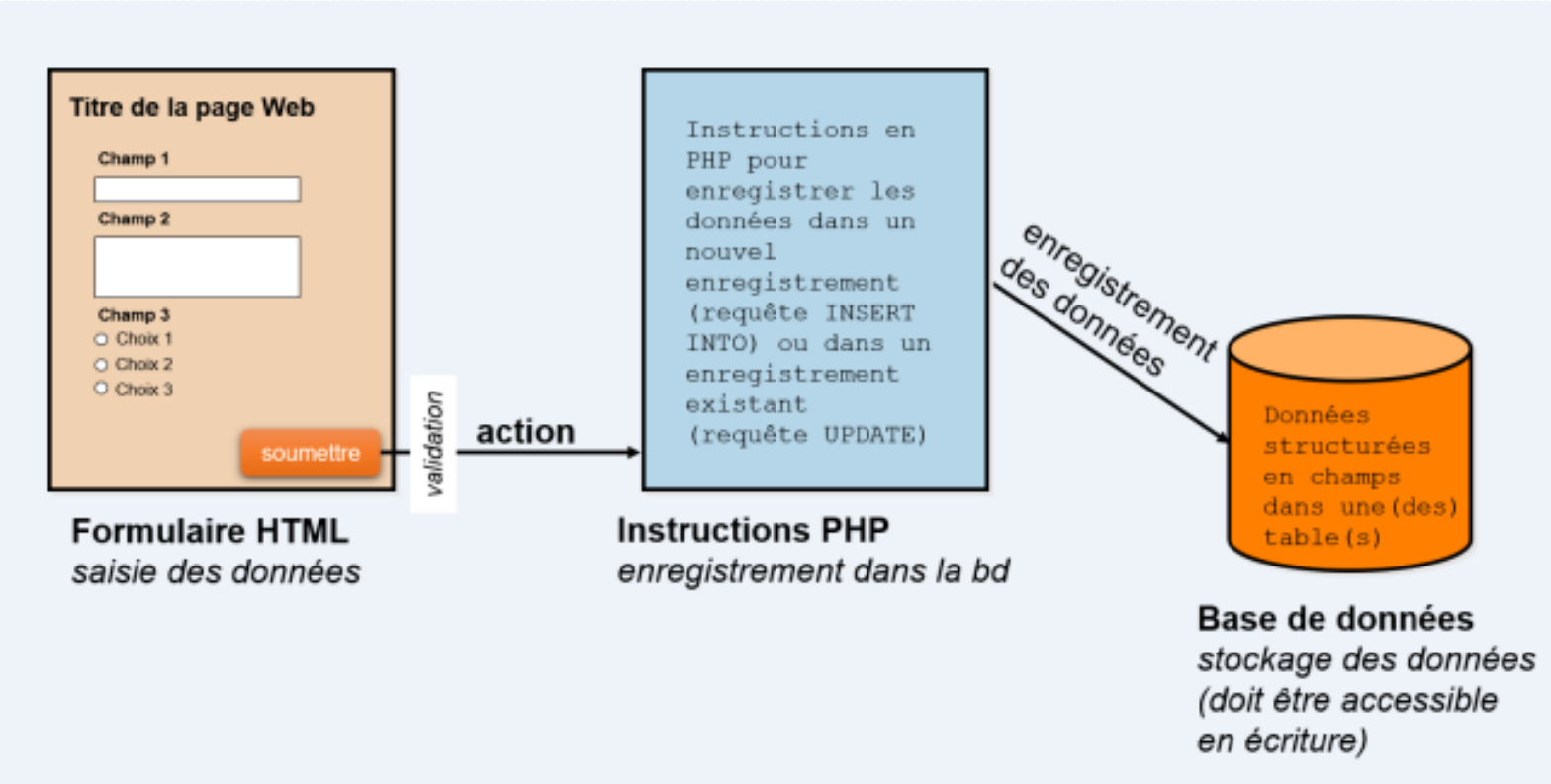
Se connecter à une base de données MySQL en PHP

Introduction

- Dans ce nouveau chapitre, nous allons passer en revue les différents moyens que nous avons de nous connecter au serveur et donc à nos bases de données MySQL en PHP.
- Dans cette partie on dois avoir une base de données prête à être utilisée dans PHP

Introduction

- Schéma explicatif:



Se connecter à MySQL en PHP : les API proposées par le PHP

- Pour se connecter à MySQL, le PHP met à notre disposition deux extensions :

1. L'extension MySQLi: Syntaxe **orientée procédure** avec des fonctions telles que mysqli_query().
2. L'extension **PDO** (PHP Data Objects): Approche **orientée objet** avec des méthodes telles que prepare() et execute().

- MySQLi ne va fonctionner qu'avec les bases de données MySQL.
- PDO va fonctionner avec d'autres systèmes de gestion des bases de données(pas obligatoirement mysql).

Les étapes à suivre pour travailler avec des données de BD

1. Se connecter au serveur de BD, et choisir le nom de la base sur laquelle on va effectuer des requêtes.
2. Définir la requête SQL (sélection par exemple)
3. Exécuter la requête SQL au niveau de la base de données.
4. Récupérer et lire les données résultante de la requête de sélection.

Connexion au serveur avec PDO

- Pour se connecter en utilisant PDO, on dois fournir les informations concernant:
  1. **Nom du serveur**
  2. **Nom de la base de données**
  3. **Nom d'utilisateur**
  4. **Un mot de passe.**

Syntaxe:

```
$conn = new PDO(' mysql : host=NomServeur ; dbname=NomBase ' ,
                                    'Utilisateur', ' mot de passe');
```

Connexion au serveur avec PDO

- Exemple:

```php
<?php

$NomServeur = 'localhost';
$Utilisateur= 'root';
$MotDePasse= '';

//On établit la connexion
$conn = new PDO("mysql:host= $NomServeur ; dbname=bddtest",
$Utilisateur , $MotDePasse);

?>
```

Tester la présence d'erreurs

- S'il y a une erreur de connexion à la base de données, PHP va afficher toute la ligne qui pose l'erreur, ce qui inclut le mot de passe, mais on ne dois pas laisser les visiteurs voir ces informations délicates. C'est pour cela qu'on va utiliser les exceptions pour attraper l'erreur à l'aide des instruction

  try{} catch(){}

Tester la présence d'erreurs

- Codage:

```
try{
$conn = new PDO('mysql:host=localhost; dbname=bddtest', 'root', '');
}
catch (Exception $e)
{
        die('Erreur : ' . $e->getMessage());
}
```

- PHP essai d'exécuter les instructions (try), s'il 'a une erreur, il rentre dans le bloc catch.
- **die** est une instruction qui permet d'arrêter l'exécution et afficher un message d'erreur.

Lire les données de la base

- Dans cette partie on va voir comment lire des données à partir d'une base de données, c'est-à-dire comment sélectionner des données. Ou bien comment exécuter une instruction SQL SELECT à partir de PHP.

Exécuter une requête de sélection avec PHP

- Pour exécuter une requête SELECT, il faut se servir de la méthode **query** de la manière suivante:
- syntaxe:

```
$resultat=$conn ->query(requêteSQL);
```

1. **$conn**: c'est l'objet PDO qui représente la connexion à la base de données (voir les diapositifs précédents)
2. **Query:** est une méthode de l'objet PDO utilisée pour **exécuter** une requête **SELECT**
3. **requetteSQL**: est une chaine de caractère qui contient une requête SELECT
4. **$resultat**: représente les données récupérés.

Exécuter une requête de sélection avec PHP

- Exemple de code:

```
$maRequete =" SELECT * FROM stagiaires ";

$resultat=$conn ->query($maRequete);
```

Remarque: on peut écrire notre requête SELECT directement entre parenthèses, sans avoir besoin de la définir dans une variable $maRequette.

Lire des données récupérées

- Il existes des méthodes pour lire les données récupérées:

1. **fetch():** S'ouvre comme un pointeur prêt à parcourir les données une par une, c'est-à-dire ligne par ligne (Généralement à travers une boucle **while**). C'est la méthode recommandée lorsque vous attendez beaucoup de données. S'il n y'a aucune ligne à lire elle retourne false.
2. **fetchAll()**: Apporte toutes les données à la fois, sans ouvrir de pointeur, en les stockant dans un **tableau**. Elle est recommandé lorsque vous ne attendez pas trop de résultats qui pourraient causer des problèmes de mémoire,

Lire des données récupérées: méthode fetch()

- Exemple 1 de code:

```
try{

$conn=new PDO('mysql:host=localhost;dbname=gestion stagiaires','root','');

}
catch(Exception $x){
        die('erreur de connexion :'.$x->getMessage());
}

$result=$conn->query("select * from stagiaires ");

while($ligne=$result->fetch()){

        echo $ligne['nom']." ".$ligne['prenom']."<br>";
}
```

Lire des données récupérées: méthode fetch()

- Exemple 2 de code:

```
try{

$conn=new PDO('mysql:host=localhost;dbname=gestion stagiaires','root','');

}
catch(Exception $x){
        die('erreur de connexion :'.$x->getMessage());
}

$result=$conn->query("select * from stagiaires ");

while($ligne=$result->fetch()){

        echo $ligne[1]." ".$ligne[2]."<br>";
}
```

Lire des données récupérées: méthode fetchAll()

- La méthode fetchAll() retourne un tableau contenant le résultat de la requête. Elle peut être utilisée sans paramètres Ou avec un paramètre qui s'appelle **fecth_style:**
- **Exemple de code:**

```
try{
$conn=new PDO('mysql:host=localhost;dbname=gestion stagiaires','root','');
}
catch(Exception $x){
                die('erreur de connexion :'.$x->getMessage());
}

$result=$conn->query("select * from stagiaires ");
$TabRes= $result->fetchAll();

foreach ($TabRes as  $ligne)
{
echo $ligne['nom']." ".$ligne['prenom']."<br>";

}
```

- **Remarque: on peut utiliser également l'indice de la colonne au lieu du nom de la colonne.**

Les méthodes fetch() et fetchAll() fetch_style

- La méthode fetchAll() peut accepter un paramètre qui désigne le style de tableau de résulats.

1. **PDO::FETCH_ASSOC**: signifie que le tableau de résultat est un tableau associatif. Dans ce cas les clés de ce tableau sont les noms des colonnes dans la requête SELECT.
2. **PDO::FETCH_NUM:** signifie que le tableau de résultat est un tableau numérique. Dans ce cas les clés de ce tableau sont les indices des colonnes dans la requête SELECT.
3. **PDO::FETCH_BOTH:** signifie qu'on va pouvoir à la fois utiliser le tableau de résultats soit comme un tableau associatif ou numérique. C'est la valeur par défaut.

La méthode fetchAll() fecth_style

- Exemple:

```
try{
$conn=new PDO('mysql:host=localhost;dbname=gestion stagiaires','root','');
}
catch(Exception $x){
                die('erreur de connexion :'.$x->getMessage());
}

$result=$conn->query("select * from stagiaires ");
$TabRes= $result->fetchAll(PDO::FETCH_ASSOC);

foreach ($TabRes  as  $ligne)
{
echo $ligne['nom']." ".$ligne['prenom']."<br>";

}
```

La méthode fetchAll() fecth_style

- Exemple:

```
try{
$conn=new PDO('mysql:host=localhost;dbname=gestion stagiaires','root','');
}
catch(Exception $x){
                die('erreur de connexion :'.$x->getMessage());
}

$result=$conn->query("select * from stagiaires ");
$TabRes= $result->fetchAll(PDO::FETCH_NUM);

foreach ($TabRes  as  $ligne)
{
echo $ligne[0]." ".$ligne[1]."<br>";

}
```

Compter le nombre de lignes récupérées

- Pour connaitre le nombre de ligne récupérées, on peut se servir de la méthode rowCount() de la manière suivante:

```
$nb=$result ->rowCount();
```

- La variable $nb est un entier. C'est le nombre de ligne lues par la requête select.

Compter le nombre de lignes récupérées

- Pour connaitre le nombre de colonnes récupérées on peut se servir de la méthode columnCount() de la manière suivante:

```
$nb=$resultat->columnCount();
```

- La variable $nb est un entier. C'est le nombre de colonnes lues par la requête select.

Libère la connexion au serveur

- La méthode **closeCursor()** libère la connexion au serveur, c'est-à-dire libère la mémoire pour les variables qui contiennent les requêtes
- permettant ainsi à d'autres requêtes SQL d'être exécutées, mais laisse la requête dans un état lui permettant d'être de nouveau exécutée.

- Syntaxe:

```
$result = $conn->query($Marequête);
…
$result->closeCursor();
```

Fermer la connexion

- Pour fermer la connexion, vous devez **détruire** l'objet en vous assurant que toutes ses références sont effacées.
- Vous pouvez faire cela en assignant **null** à la variable gérant l'objet.

```
$conn = null;
```

- Si vous ne le faites pas explicitement, PHP fermera automatiquement la connexion lorsque le script arrivera à la fin.