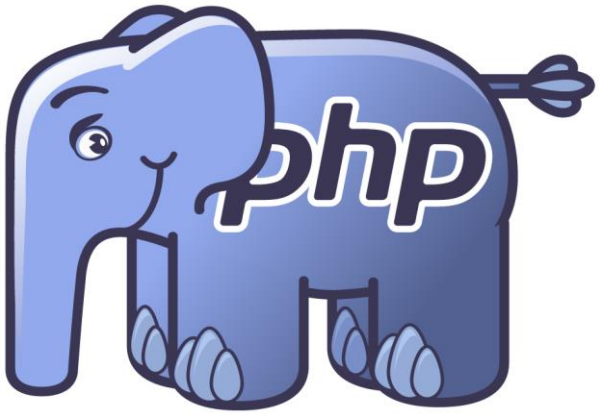
Chapitre 4. PHP-Partie 2-

Prof. Amina ADADI

2ème Année Génie Logiciel

Année universitaire 2019-2020

Persistance: Session & Cookies



COOKIES

Un cookie est un fichier petit (65 Ko maxi) que l'on stocke sur le poste du visiteur.

Exemple :

Vous stockez le pseudo du visiteur. Lorsqu'il revient sur votre site, vous pouvez lire dans le cookie et lui afficher "Bonjour votre_pseudo".

Contrairement à certaines légendes, les cookies sont totalement inoffensifs, ce ne sont pas des virus ou troyens ou je ne sais quoi !

COOKIES

Création des cookies

On utilise la fonction setcookie()

```php
<?php
   /* on définit une durée de vie de notre cookie (en
   secondes), donc un an dans notre cas*/
   $temps = 365*24*3600;
   /* on envoie un cookie de nom pseudo portant la valeur
   Belaid*/
   setcookie ("pseudo", "Belaid", time() + $temps);
?>
```

COOKIES

Attention

Plusieurs conditions sont à respecter afin que l'utilisation des cookies se passe au mieux :

1. l'envoie d'un cookie doit être la première fonction PHP que vous utilisez dans votre script,
   ce qui veut dire que vous devez utiliser la fonction setcookie() tout en haut de votre script (AUCUN AFFICHAGE ET AUCUN CODE HTML AVANT UN SETCOOKIE)
2. Si d'autres fonctions interviennent avant l'envoi du cookie, celui-ci ne fonctionnera pas.
3. Si vous envoyez un cookie sur un poste client, celui-ci effacera automatiquement l'ancien cookie qui portait le même nom (si il y en avait un), autrement il le créera
4. Note : pour effacer un cookie, vous devez lancer un cookie qui aura le même nom que le cookie que vous voulez effacer, tout en lui donnant une valeur nulle (vous pouvez également l'envoyer avec un temps de vie dépassé).

COOKIES

Afficher un cookie

```
$_COOKIE['Nom_cookie']
```

```php
<?php
echo "Bonjour " .$_COOKIE['Nom'] ." " .$_COOKIE['Prenom']; // Afficher les cookies

// Ancienne syntaxe pour afficher un cookie, elle est obsoléte
echo $HTTP_COOKIE_VARS["Nom"];
?>
```

Modifier un cookie

```php
<?php
setcookie('Prenom', 'Ali', $timestamp);
?>
```

COOKIES

Supprimer un cookie

Pour supprimer un cookie , on utilise **setcookie()** avec une date d'expiration passée:

```php
<?php
// la date d'expiration est une heure avant
setcookie("user", "", time() - 3600);

?>
```

Vérifier si les cookies sont « enable »

```php
<?php
setcookie("test_cookie", "test", time() + 3600, '/');
?>
<html><body>
<?php
if(count($_COOKIE) > 0) {
    echo "Cookies activés.";
} else {
    echo "Cookies désactvé.";
}
?>
```

COOKIES

Exemple: **Sondage avec vérification des votes**

```php
//1er partie du script PHP
if(isset($_POST["choix"]))
{
if($_COOKIE["votant"] ←(1) && $_COOKIE["vote"]) ←(2)
){
$vote = $_COOKIE["vote"];
?>
<!--Code JavaScript -->
<script type="text/javascript" >
alert('Vous avez déja voté pour <?php echo $vote ?>!') ← (3)
</script>
<!-- 2 eme partie du script PHP-->
<?php
}
else
{
$vote = $_COOKIE["vote"];
setcookie("votant","true",time()+300);
setcookie("vote","{$_POST['choix']}",time()+300);
//enregistrement du vote dans un fichier --> voir chapitre 11
if(file_exists("sondage.txt") )
{if($id_file=fopen("sondage.txt","a"))
{
flock($id_file,2);
fwrite($id_file,$_POST['choix']."\n");
flock($id_file,3);
fclose($id_file);
}
else
{ echo "Fichier inaccessible";}
}
else
{
$id_file=fopen("sondage.txt","w");
fwrite($id_file,$_POST['choix']."\n");
fclose($id_file);
}
// Fin de l'enregistrement
?>
```

COOKIES

Exemple: **Sondage avec vérification des votes**

```
<!--Code JavaScript -->
<script type="text/javascript" >
alert('Merci de votre vote pour <?php echo $_POST["choix"] ?> ! ')
</script>
<!-- 3 eme partie du script PHP-->
<?php
}
}
?>
<!DOCTYPE html>
<html>
<head><title>Sondage </title></head>
<body>
<h2>Bienvenue sur le site PHP 5 </h2>

<form method="post" action="<?php echo $_SERVER['PHP_SELF'] ?>">
<fieldset> <legend>Votez pour votre technologie Internet préférée</legend>
<table><tbody><tr>
<td>Choix 1 : PHP/MySQL</td>
<td><input type="radio" name="choix" value="PHP et MySQL" /></td></tr>
<tr><td>Choix 2 : ASP.Net</td>
<td><input type="radio" name="choix" value="ASP.Net" /></td></tr>
<tr><td>Choix 3 : JSP </td>
<td><input type="radio" name="choix" value="JSP" /></td></tr>
<tr><td><b>Votez ! </b></td>
<td><input type="submit" value="ENVOI" /></td></tr>
</tbody></table></fieldset></form></body></html>
```

SESSION

Intérêt

- Aller plus loin que les cookies
- Permettre à plusieurs pages de partager les mêmes données
- Conserver plusieurs données lors d'une session
- Ranger les données de la session sur le serveur
- Sécuriser le passage des données vers le serveur en
  - Donnant un identifiant à la session
  - Cryptant l'identifiant
- Disposer de plusieurs sessions et donner un nom à chacune

SESSION

Intérêt

- Aller plus loin que les cookies
- Permettre à plusieurs pages de partager les mêmes données
- Conserver plusieurs données lors d'une session
- Ranger les données de la session sur le serveur
- Sécuriser le passage des données vers le serveur en
  - Donnant un identifiant à la session
  - Cryptant l'identifiant
- Disposer de plusieurs sessions et donner un nom à chacune

SESSION

- Les sessions en PHP permettent de sauvegarder des variables de **page en page** pendant une certaine durée prédéfinie par PHP (et modifiable bien entendu).
- Chaque utilisateur ayant besoin des sessions se voit attribuer un identifiant unique appelé **ID de session**. Cet identifiant est stocké sur le poste de l'internaute sous forme d'un cookie ou transite via l'URL si l'option **session.use_trans_sid** est à 1 (ou On) dans le fichier **php.ini**.

SESSION

Démarrer une session

Avant d'utiliser les sessions sur une page, on doit toujours utiliser la fonction **session_start()** placée avant tout envoi de code HTML, et donc généralement tout en haut de votre page PHP :

```
session_start()
```

Création une variable de session

```
$_SESSION['nom']="Valeur"
```

SESSION

- **Exemple**

```php
<?php
session_start(); // On démarre la session AVANT toute chose,
// On crée dans cette exemple 3 variables de session :
$_SESSION['prenom'] = 'Ahmad';
$_SESSION['nom'] = 'Ahmadi';
$_SESSION['age'] = 20;
// Maintenant on peut taper du code HTML
?>
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Variables de session</title>
<body>
<p>Bonjour <?php echo $_SESSION['prenom']; ?>!</p>>
<p>Tu es à l accueil de mon site </p>
</body>
</html>
```

SESSION

Durée de vie d'une session

➢Dés que l'on ferme le navigateur la session est détruite.
Sauf à configurer le fichier **php.ini** avec session.lifetime qui fixe la durée de vie, en secondes, du cookie envoyé au client. La valeur 0 signifie "jusqu'à ce que le client soit fermé". Par défaut à 0.
Donc si on le laisse à zéro la session dure jusqu'à la fermeture du navigateur, pour laisser les données durant 30 minutes, il faut remplacer 0 par 1800 (= 30 minutes * 60 secondes dans une minute). session.lifetime = 0

SESSION

Affichage id session

➢Lit et/ou modifie l'identifiant courant de session

```
echo session_id()
```

Suppression d'une variable de session

```
boolean session_unregister (string name)
```

- La commande **session_unregister()** supprime une variable dans la session courante.
- Elle retourne TRUE si success, FALSE sinon.

```
session_unset()
```

- Il est aussi possible de purger toutes les variables de la session avec

```
unset($_SESSION["mavariable"])
```

- Si vous utilisez le tableau superglobale **$_SESSION**, il suffit alors d'utiliser **unset**()

SESSION

Fermeture d'une session

```
session_write_close();

session_destroy();
```

Fermeture conservatrice avec session_write_close

- La commande **session_write_close()** écrit les valeurs des variables de session sur le serveur et ferme la session.

Fermeture destructive avec session_destroy

Si vous voulez détruire la session du visiteur, vous pouvez faire un lien "Déconnexion" qui améne vers une page qui fait appel à la fonction session_destroy()
La commande **session_destroy()** détruit toutes les données enregistrées d'une session.
Cette derniére commande est la plus utilisée car n'impose aucune sauvegarde au serveur.
Retourne TRUE en cas de succés, et FALSE sinon.

SESSION

Exemple:

Commande en ligne

```php
<?php
session_start();
//AJOUTER
if($_POST["envoi"]=="AJOUTER" && $_POST["code"]!="" &&
$_POST["article"]!="" && $_POST["prix"]!="")
{
$code=$_POST["code"];
$article= $_POST["article"];
$prix= $_POST["prix"];
$_SESSION['code']= $_SESSION['code']."//".$code;
$_SESSION['article']= $_SESSION['article']."//".$article;
$_SESSION['prix']= $_SESSION['prix']."//".$prix;
}
//VERIFIER
if($_POST["envoi"]=="VERIFIER")
{
echo "<table border=\"1\" >";
echo "<tr><td colspan=\"3\"><b>Récapitulatif de votre commande</b></td>";
echo "<tr><th> code </th><th> article </ th><th> 
prix </th>";
$total=0;
$tab_code=explode("//",$_SESSION['code']);
$tab_article=explode("//",$_SESSION['article']);
$tab_prix=explode("//",$_SESSION['prix']);
for($i=1;$i<count($tab_code);$i++)
{
echo "<tr> <td>{$tab_code[$i]}</td> <td>{$tab_article[$i]}</td><td>
".sprintf("%01.2f", $tab_prix[$i])."</td>";
$prixtotal+=$tab_prix[$i];
}
echo "<tr> <td colspan=2> PRIX TOTAL </td> <td>".sprintf("%01.2f", $prixtotal)."
</td>";
echo "</table>";
}
```

SESSION

Exemple: Commande en ligne

```
//ENREGISTRER
if($_POST["envoi"]=="ENREGISTRER")
    {
$idfile=fopen("commande.txt",w);
//
$tab_code=explode("//",$_SESSION['code']);
$tab_article=explode("//",$_SESSION['article']);
$tab_prix=explode("//",$_SESSION['prix']);
for($i=0;$i<count($tab_code);$i++) ←

{
fwrite($idfile, $tab_code[$i]." ; ".$tab_article[$i]." ; ".$tab_prix[$i].";
\n");
}
fclose($idfile);
}
//LOGOUT
if($_POST["envoi"]=="LOGOUT")
{
session_unset();
session_destroy();
echo "<h3>La session est terminée</h3>";
}
$_POST["envoi"]="";
?>
```

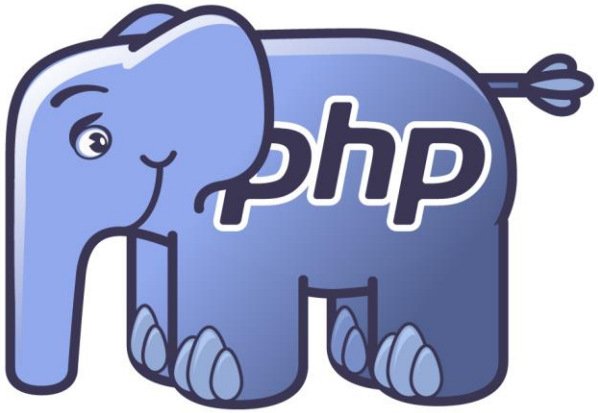
SESSION

Exemple:

Commande en ligne

```
<!DOCTYPE >
<html>
<head>
 <title>Gestion de panier</title>
</head>
<body>
 <form action="<?php $_SERVER['PHP_SELF'] ?>" method="post"
enctype="application/x-www-form-urlencoded">
<fieldset>
 <legend><b>Saisies d'articles</b></legend>
<table>
<tbody>
<tr>
 <th>code : </th>
 <td> <input type="text" name="code" /></td>
</tr>
<tr>
 <th>article : </th>
 <td><input type="text" name="article" /></td>
</tr>
<tr>
 <th>prix :</th>
 <td><input type="text" name="prix" /></td>
</tr>
<tr>
<td colspan="3">
 <input type="submit" name="envoi" value="AJOUTER" />
 <input type="submit" name="envoi" value="VERIFIER" />
 <input type="submit" name="envoi" value="ENREGISTRER" />
 <input type="submit" name="envoi" value="LOGOUT" />
</td>
</tr>
</tbody>
```

Accès à la base de données

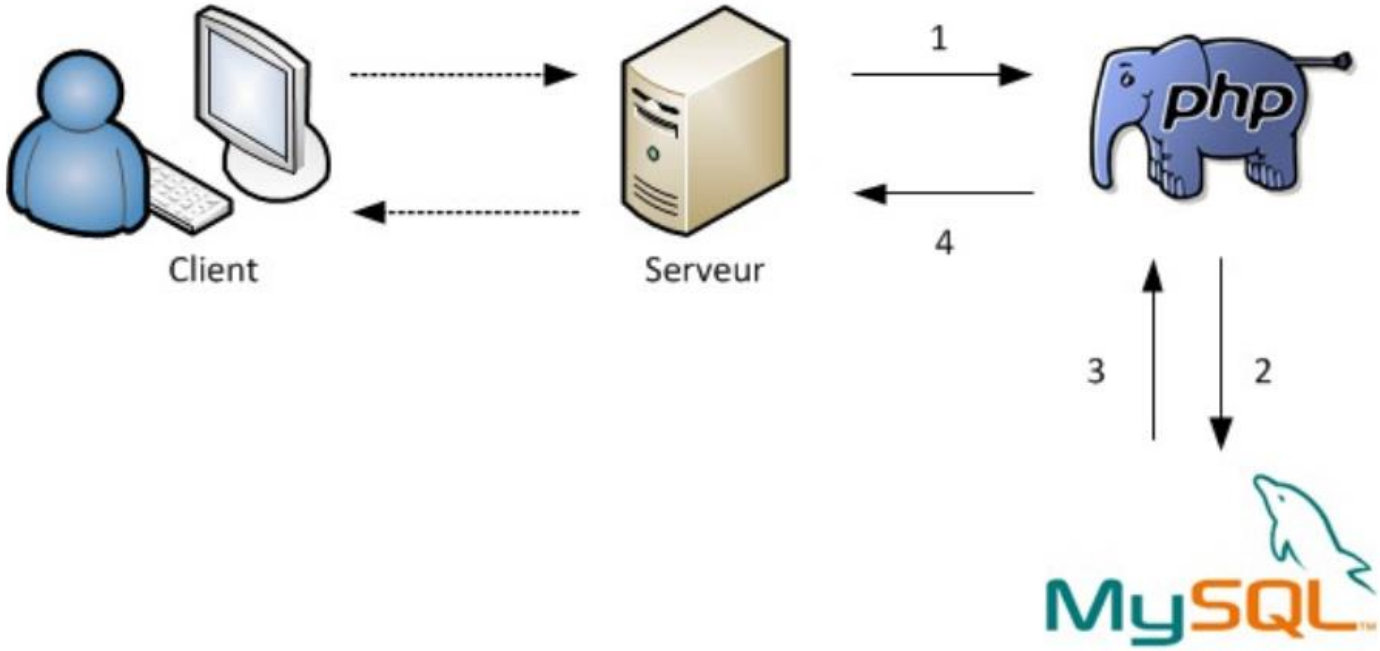


3 façons pour accéder à la BD

PHP propose plusieurs moyens de se connecter à une base de données MySQL.

- **L'extension mysql_** : ce sont des fonctions qui permettent d'accéder à une base de données MySQL et donc de communiquer avec MySQL. Leur nom commence toujours par mysql_. Toutefois, ces fonctions sont vieilles et on recommande de ne plus les utiliser aujourd'hui.
- **•L'extension mysqli_** : ce sont des fonctions améliorées d'accès à MySQL (Le « i » pour « improved »).Elles proposent plus de fonctionnalités et sont plus à jour.il y'a deux variation: MySQLi (object-oriented) et MySQLi (procedural)
- **•L'extension PDO :** c'est un outil complet qui permet d'accéder à n'importe quel type de base de données. On peut donc l'utiliser pour se connecter aussi bien à MySQL que PostgreSQL ou Oracle

Scénario de communication



1. le serveur utilise toujours PHP, il lui fait donc passer le message
2. PHP effectue les actions demandées et se rend compte qu'il a besoin de MySQL. En effet, le code PHP contient à un endroit « Va demander à MySQL d'enregistrer ce message ». Il fait donc passer le travail à MySQL ;
3. MySQL fait le travail que PHP lui avait soumis et lui répond « O.K., c'est bon ! » ;
4. PHP renvoie au serveur que MySQL a bien fait ce qui lui était demandé

Classes prédéfinies PDO

- **PDO** : connexion PHP / base de données
  - `__construct()`
  - `exec(), prepare(), query()`
  - `errorCode(), errorInfo()`
  - `getAttributes(), setAttribute()`
  - `lastInsertId(), quote()`
  - `beginTransaction()`
  - `commit(), rollBack()`
  - `getAvailableDrivers()`

Classes prédéfinies PDO

- **PDOStatement** : requête préparée, jeu de résultats
  - `bindColumn()`, `bindParam()`, `bindValue()`, `closeCursor()`
  - `errorCode()`, `errorInfo()`
  - `fetch()`, `fetchAll()`, `fetchColumn()`, `fetchObject()`, `setFetchMode()`, `nextRowset()`
  - `rowCount()`, `columnCount()`, `getColumnMeta()`
  - `getAttribute()`, `setAttribute()`
  - `execute()`
  - `debugDumpParams()`

fonctions de l'extension MySQLi

Interface POO	Interface procédural	Description
mysqli::__construct()	mysqli_connect()	Ouvre une connexion à un serveur MySQL
mysqli::select_db()	mysqli_select_db()	Sélectionne une base de données par défaut pour les requêtes
mysqli::query()	mysqli_query()	Exécute une requête sur la base de données
$mysqli_result::num_rows	mysqli_num_rows()	Le nombre de lignes dans un résultat
mysqli_result::fetch_array()	mysqli_fetch_array()	Retourne une ligne de résultat sous la forme d'un tableau associatif, d'un tableau indexé, ou les deux
mysqli::close()	mysqli_close()	Ferme une connexion

Principales opération sur la base de données

- CONNEXION
- EFFECTUER DES REQUÊTES
- LES REQUÊTES PRÉPARÉES

Connexion

MySQLi Object-Oriented

```php
<?php
$servername = "localhost";
$username = "username";
$password = "password";

// Création de connexion
$conn = new mysqli($servername, $username, $password);

// Vérification de la connexion
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
echo "Connected successfully";
?
```

Connexion

MySQLi Procedural

```php
<?php
$servername = "localhost";
$username = "username";
$password = "password";

// Création de connexion
 $conn = mysqli_connect($servername, $username,
$password);

// Vérification de la connexion
 if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}
echo "Connected successfully";
?
```

Connexion

PDO

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";

try {
    $conn = new PDO("mysql:host=$servername;dbname=myDB",
$username, $password);
    // Le mode PDO error est a réglé a EXCEPTION
    $conn->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);
    echo "Connected successfully";
    }
catch(PDOException $e)
    {
    echo "Connection failed: " . $e->getMessage();
    }
?>
```

Connexion

Fermeture de la connexion

MySQLi Object-Oriented:

```
$conn->close();
```

MySQLi Procedural:

```
mysqli_close($conn);
```

PDO:

```
$conn = null;
```

EFFECTUER DES REQUÊTES

PDO distingue 2 types de requêtes :

- les requêtes de sélection : **SELECT** > query($requete)/ mysqli_query($requete)
- les requêtes de modification insertion: **UPDATE – INSERT – DELETE** > exec($requete)/ mysqli_query($requete)

La différence entre ces deux méthodes est que query() retournera un jeu de résultats sous la forme d'un objet PDOStatement pour PDO

Alors que exec() retournera uniquement le nombre de lignes affectées. En d'autres termes, vous utiliserez query() pour des requêtes de sélection (SELECT) et exec() pour des requêtes d'insertion (INSERT), de modification (UPDATE) ou de suppression DELETE)

EFFECTUER DES REQUÊTES

<table>
<tr><th colspan="2">Requête SQL retournant une seule ligne</th></tr>
<tr><td>PDO
$query = 'SELECT * FROM salaries
WHERE
id=1;';
$ligne = $pdo->query($query)
->fetch();</td><td>mysqli
$query = 'SELECT * FROM salaries
where id=1;';
$result = mysqli_query($conn,$query);
$ligne = mysqli_fetch_assoc($result);</td></tr>
<tr><th colspan="2">Requête SQL retournant plusieurs lignes</th></tr>
<tr><td>$query = 'SELECT * FROM salaries;';
$liste = $pdo->query($query)
->fetchAll();</td><td>$query = 'SELECT * FROM salaries;';
$result = mysqli_query($conn,$query);
$liste = array();
while($ligne= mysqli_fetch_assoc($resu
lt)){
$liste[] = $ligne ;
}</td></tr>
<tr><th colspan="2">Requête d'insertion, de modification ou de suppression</th></tr>
<tr><td>$query = 'DELETE FROM salaries WHERE
id=2;';
$rowCount = $pdo->exec($query);</td><td>$query = 'DELETE FROM salaries WHERE
id=2;';
mysqli_query(sconn,$query);
$rowCount = mysq l i_affected_rows();</td></tr>
</table>

LES REQUÊTES PREPAREES

Le principe est de préparer un « moule » de requête et de placer des *place holders aux endroits où* l'on voudra insérer nos valeurs dynamiques.
Vous avez une requête avec des place holders (les points d'interrogation) :

```
$requete ='DELETE FROM salaries WHERE idsalaries = ?';
```

Le SGBD va préparer (interpréter, compiler et stocker temporairement son "plan d'exécution logique") la requête.

PDO et Mysqli

```
$stmt = $conn->prepare($requete)
```

LES REQUÊTES PREPAREES

Une fois préparé, et lors de l'exécution il faut d'abord binder les paramètre . C-à-d, Il faudra associer des valeurs aux place holders, qui agissent un peu comme des variables `$stmt->bind_param`. Ensuite on exécute la requête préparée : `$stm->execute() ;`

Mysqli

```
$stmt = $conn->prepare("INSERT INTO MyGuests
(firstname, lastname, email) VALUES (?, ?, ?)");
$stmt->bind_param("sss", $firstname, $lastname,
$email);

// affecter les paramètre et exécuter
$firstname = « ahmad";
$lastname = « ahmadi";
$email = « ahmadi@example.com";
$stmt->execute();
```

L'argument "sss" liste le type de données des paramètres. "S" veut dire string, i - integer d - double

LES REQUÊTES PREPAREES

PDO

```
 // bind parameters
    $stmt = $conn->prepare("INSERT INTO MyGuests
(firstname, lastname, email)
    VALUES (:firstname, :lastname, :email)");
    $stmt->bindParam(':firstname', $firstname);
    $stmt->bindParam(':lastname', $lastname);
    $stmt->bindParam(':email', $email);

    // exécuter
  $firstname = « ahmad";
$lastname = « ahmadi";
$email = « ahmadi@example.com";
    $stmt->execute();
```

LES REQUÊTES PREPAREES

Exemple complet PDO

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDBPDO";

try {
    $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username,
$password);
    // set the PDO error mode to exception
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

    // prepare sql and bind parameters
    $stmt = $conn->prepare("INSERT INTO MyGuests (firstname, lastname, email)
    VALUES (:firstname, :lastname, :email)");
    $stmt->bindParam(':firstname', $firstname);
    $stmt->bindParam(':lastname', $lastname);
    $stmt->bindParam(':email', $email);
```

LES REQUÊTES PREPAREES

Exemple complet PDO

```
    // insert a row
    $firstname = " Meryem ";
    $lastname = " ALAMI ";
    $email = " Meryem @example.com";
    $stmt->execute();

    // insert another row
    $firstname = "salim";
    $lastname = "slimi";
    $email = " salim @example.com";
    $stmt->execute();

    echo "New records created successfully";
    }
catch(PDOException $e)
    {
    echo "Error: " . $e->getMessage();
    }
$conn = null;
?>
```

LES REQUÊTES PREPAREES

Exemple complet mysqli

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

// prepare and bind
$stmt = $conn->prepare("INSERT INTO MyGuests (firstname, lastname, email) VALUES (?,
?, ?)");
$stmt->bind_param("sss", $firstname, $lastname, $email);
```

LES REQUÊTES PREPAREES

Exemple complet mysqli

```
// set parameters and execute
$firstname = "John";
$lastname = "Doe";
$email = "john@example.com";
$stmt->execute();

$firstname = "Mary";
$lastname = "Moe";
$email = "mary@example.com";
$stmt->execute();

$firstname = "Julie";
$lastname = "Dooley";
$email = "julie@example.com";
$stmt->execute();

echo "New records created successfully";

$stmt->close();
$conn->close();
?>
```

Pour aller plus loin dans PHP

- Accès au système de fichier
- Gestion des dates calendrier
- Gestion dynamique des images
- Gestion des chaines de caractère et expression régulière