PHP and MySQL Continued

Course Code:CSC 3222 Course Title: Web Technologies

Dept. of Computer Science
Faculty of Science and Technology

Lecturer No:	**08**	**Week No:**	**08**	**Semester:**	
Lecturer:	*Sazzad Hossain* *sazzad@aiub.edu*				

Lecture Outline

1. Prepared Statement And Bound Parameters
2. Select Query
3. Filtering data
4. Update and Delete Query
5. Insert/Upload Images or files

Prepared Statement

Topic sub heading..

The MySQL database supports prepared statements.

A prepared statement is a feature used to execute the same (or similar) SQL statements repeatedly with high efficiency.

Prepared statements are very useful against SQL injections.

Prepared Statement

Work flow

Prepared statements basically work like this:

1. Prepare: An SQL statement template is created and sent to the database. Certain values are left unspecified, called parameters (labeled "?"). Example: INSERT INTO MyGuests VALUES(?, ?, ?)
2. The database parses, compiles, and performs query optimization on the SQL statement template, and stores the result without executing it
3. Execute: At a later time, the application binds the values to the parameters, and the database executes the statement. The application may execute the statement as many times as it wants with different values

Prepared Statement

Advantages

Prepared statements have three main advantages:

1. Prepared statements reduce parsing time as the preparation on the query is done only once (although the statement is executed multiple times)
2. Bound parameters minimize bandwidth to the server as you need send only the parameters each time, and not the whole query
3. Prepared statements are very useful against SQL injections, because parameter values, which are transmitted later using a different protocol, need not be correctly escaped. If the original statement template is not derived from external input, SQL injection cannot occur.

Example

```
<?php
$servername = "localhost";
$username = “root";
$password = "";
$dbname = "myDB";
$conn = new mysqli($servername, $username, $password, $dbname);
if ($conn->connect_error) {    die("Connection failed: " . $conn->connect_error);}
$stmt = $conn->prepare("INSERT INTO Users (firstname, lastname, email) VALUES (?, ?, ?)"); // prepare and bind
$stmt->bind_param("sss", $firstname, $lastname, $email);
// 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();
?>
```

Explanation

"INSERT INTO MyGuests (firstname, lastname, email) VALUES (?, ?, ?)"
question mark (?) represents the substitute of an integer, string, double or blob value data type.

bind_param() function:
$stmt->bind_param("sss", $firstname, $lastname, $email);
This function binds the parameters to the SQL query and tells the database what the parameters are. The "sss" argument lists the types of data that the parameters are. The s character tells mysql that the parameter is a string.

Explanation

The argument may be one of four types:

i - integer
d - double
s - string
b - BLOB
must have one of these for each parameter.

This is to minimize the risk of SQL injections.

Select Query

- The SELECT statement is used to select data from one or more tables:

SELECT column_name(s) FROM table_name

- the * character to select ALL columns from a table:

SELECT * FROM table_name

example selects the id, firstname and lastname columns from the Users table and displays it

Example

```
<?php
$servername = "localhost";
$username = “root";
$password = "";
$dbname = "myDB";
$conn = new mysqli($servername, $username, $password, $dbname);
if ($conn->connect_error) { die("Connection failed: " . $conn->connect_error);}
$sql = "SELECT id, firstname, lastname FROM Users";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
   // output data of each row
   while($row = $result->fetch_assoc()) {
      echo "id: " . $row["id"]. " - Name: " . $row["firstname"]. " " . $row["lastname"]. "<br>";
   }
} else {
   echo "0 results";
}
$conn->close();
?>
```

Explanation

- SQL query selects the id, firstname and lastname columns from the Users table.

- The query puts the resulting data into a variable called $result.

- The function num_rows() checks if there are more than zero rows returned.

- If there are more than zero rows returned, the function fetch_assoc() puts all the results into an associative array that can loop through. The while() loop loops through the result set and outputs the data from the id, firstname and lastname columns.

Filtering data

Using Where Query

- The WHERE clause is used to filter records.
- The WHERE clause is used to extract only those records that fulfill a specified condition.

```
SELECT column_name(s) FROM table_name WHERE
column_name operator value
```

Example

```
<?php
$servername = "localhost";
$username = “root";
$password = "";
$dbname = "myDB";
$conn = new mysqli($servername, $username, $password, $dbname);
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
$sql = "SELECT id, firstname, lastname FROM Users WHERE lastname='Doe'";
$result = $conn->query($sql);
if ($result->num_rows > 0) {
    while($row = $result->fetch_assoc()) {
        echo "id: " . $row["id"]. " - Name: " . $row["firstname"]. " " .
$row["lastname"]. "<br>";
    }
} else {
    echo "0 results";
}
$conn->close();
?>
```

Explanation

- SQL query selects the id, firstname and lastname columns from the Users table where the lastname is "Doe". The next line of code runs the query and puts the resulting data into a variable called $result.

- The function num_rows() checks if there are more than zero rows returned.

- If there are more than zero rows returned, the function fetch_assoc() puts all the results into an associative array that can loop through. The while() loop loops through the result set and outputs the data from the id, firstname and lastname columns.

Order Data

- The ORDER BY clause is used to sort the result-set in ascending or descending order.

- The ORDER BY clause sorts the records in ascending order by default.
- To sort the records in descending order, use the DESC keyword.

```
SELECT column_name(s) FROM table_name ORDER BY
column_name(s) ASC|DESC
```

Example

```
<?php
$servername = "localhost";
$username = “root";
$password = "";
$dbname = "myDB";
$conn = new mysqli($servername, $username, $password, $dbname);
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
$sql = "SELECT id, firstname, lastname FROM Users ORDER BY lastname";
$result = $conn->query($sql);
if ($result->num_rows > 0) {
while($row = $result->fetch_assoc()) {
        echo "id: " . $row["id"]. " - Name: " . $row["firstname"]. " " . $row["lastname"].
"<br>";
    }
} else {
    echo "0 results";
}
$conn->close();
?>
```

Explanation

- SQL query selects the id, firstname and lastname columns from the Users table. The records will be ordered by the lastname column. The next line of code runs the query and puts the resulting data into a variable called $result.

- The function num_rows() checks if there are more than zero rows returned.

- If there are more than zero rows returned, the function fetch_assoc() puts all the results into an associative array that we can loop through. The while() loop loops through the result set and outputs the data from the id, firstname and lastname columns.

Limit Data

- MySQL provides a LIMIT clause that is used to specify the number of records to return.
- The LIMIT clause makes it easy to code multi page results or pagination with SQL, and is very useful on large tables. Returning a large number of records can impact on performance.
- Assume we wish to select all records from 1 - 30 (inclusive) from a table called "Orders".

```
$sql = "SELECT * FROM Orders LIMIT 30";
```

Limit Data

- Mysql provides a way to to select records 16 – 25 by using OFFSET.
- The SQL query below says "return only 10 records, start on record 16 (OFFSET 15)":

```
$sql = "SELECT * FROM Orders LIMIT 10 OFFSET 15";
```

  - shorter syntax to achieve the same result:

```
$sql = "SELECT * FROM Orders LIMIT 15, 10";
```

- the numbers are reversed when a comma is used.

Update Data

- The UPDATE statement is used to update existing records in a table:

```
UPDATE table_name
SET column1=value, column2=value2,...
WHERE some_column=some_value
```

- The WHERE clause in the UPDATE syntax
- The WHERE clause specifies which record or records that should be updated.
- If the WHERE clause is omitted, all records will be updated!

Example

```
<?php
$servername = "localhost";
$username = “root";
$password = "";
$dbname = "myDB";
$conn = new mysqli($servername, $username, $password, $dbname);
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
//query to update the record of id=2 in the “Users" table:
$sql = "UPDATE Users SET lastname='Doe' WHERE id=2";
if ($conn->query($sql) === TRUE) {
    echo "Record updated successfully";
} else {
    echo "Error updating record: " . $conn->error;
}
$conn->close();
?>
```

Delete Data

- The DELETE statement is used to delete records from a table:

```
DELETE FROM table_name
WHERE some_column = some_value
```

- The WHERE clause specifies which record or records that should be deleted.
- If the WHERE clause is omitted, all records will be deleted.

Example

```
<?php
$servername = "localhost";
$username = “root";
$password = "";
$dbname = "myDB";
$conn = new mysqli($servername, $username, $password, $dbname);
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
$sql = "DELETE FROM Users WHERE id=3";

if ($conn->query($sql) === TRUE) {
    echo "Record deleted successfully";
} else {
    echo "Error deleting record: " . $conn->error;
}

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

Insert Image into Database

Using BOLB type data

- BLOB is a kind of MySQL datatype referred as Binary Large Objects. As its name, it is used to store huge volume of data as binary strings as similar as MYSQL BINARY and VARBINARY types.

MySQL BLOB Types	Maximum Storage Length (in bytes)
TINYBLOB	((2^8)-1)
BLOB	((2^16)-1)
MEDIUMBLOB	((2^24)-1)
LONGBLOB	((2^32)-1)

Example: Insert Image

```
<HTML>
<body>
 <form enctype="multipart/form-data" action=""  method="post" >
      <label>Upload Image File:</label><br />
<input name="userImage" type="file"/>
<input type="submit"    value="Submit" />
   </form>
</body>
</HTML>
<?php
if(isset($_POST["submit"])){
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "myDB";
$conn = new mysqli($servername, $username, $password, $dbname);
if ($conn->connect_error) {die("Connection failed: " . $conn->connect_error);}
if (is_uploaded_file($_FILES['userImage']['tmp_name'])) {
      $imgData = addslashes(file_get_contents($_FILES['userImage']['tmp_name']));
      $imageProperties = getimageSize($_FILES['userImage']['tmp_name']);
      $sql = "INSERT INTO Users(imageType ,imageData) VALUES('{$imageProperties['mime']}', '{$imgData}')";
if ($conn->query($sql) === TRUE) {
   echo "image inserted successfully";
} else {
   echo "Error updating record: " . $conn->error;
}
$conn->close();
 }
}
?>
```

Explanation

The enctype attribute specifies how the form-data should be encoded when submitting it to the server. The enctype attribute can be used only if method="post".

Syntax

<form enctype="value">

Attribute Values

Value	Description
application/x-www-form-urlencoded	Default. All characters are encoded before sent (spaces are converted to "+" symbols, and special characters are converted to ASCII HEX values)
multipart/form-data	No characters are encoded. This value is required when you are using forms that have a file upload control
text/plain	Spaces are converted to "+" symbols, but no special characters are encoded

Explanation

is_uploaded_file (string $filename) : bool

- It is PHP function.
- Returns TRUE on success or FALSE on failure.
- Returns TRUE if the file named by filename was uploaded via HTTP POST.
- This is useful to help ensure that a malicious user hasn't tried to trick the script into working on files upon which it should not be working.
- For proper working, the function is_uploaded_file() needs an argument like $_FILES['userfile']['tmp_name'], - the name of the uploaded file on the client's machine $_FILES['userfile']['name'] does not work.

addslashes($string)

- The addslashes() function accepts only one parameter $string which specifies the input string which is needed to be escaped.
- It returns the escaped string with backslashes in front of the pre-defined characters which is passed in the parameter.

Examples:
Input : $string = "Geek's"
Output : Geek\'s
Input : $string='twinkle loves "coding"'
Output : twinkle loves \"coding\"

Explanation

getimagesize($filename, $image_info)

- The getimagesize() function in PHP is an inbuilt function
- used to get the size of an image.
- This function accepts the filename as a parameter and determines the image size
- returns the dimensions with the file type and height/width of image.

Example:

```
<?php
$image_info = getimagesize(“myimage.png");
print_r($image_info);
?>
```

Result:

```
Array ( [0] => 667
        [1] => 184
        [2] => 3
        [3] => width="667" height="184"
        [bits] => 8
        [mime] => image/png )
```

Example: Retrieve Image

```
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "myDB";
$conn = new mysqli($servername, $username, $password, $dbname);
if ($conn->connect_error) {die("Connection failed: " . $conn->connect_error);}

 $sql = "SELECT imageType,imageData FROM Users WHERE Id=3";
$result = $conn->query($sql);
if ($result->num_rows > 0) {
 while($row = $result->fetch_assoc()) {
header("Content-type: " . $row["imageType"]);
echo $row["imageData"];
}
}
else {echo "0 results";}
$conn->close();
?>
```

References

- MySQL - www.mysql.com
- W3Schools Online Web Tutorials- www.w3schools.com
- PHP Manual - www.php.net

Books

- Sams Teach Yourself Ajax JavaScript and PHP All in One; Phil Ballard and Michael Moncur;
- Sams Publishing; 2010
- JavaScript Phrasebook; Christian Wenz; Sams Publishing; 2007
- PHP and MySQL Web Development, 4/E; Luke Welling and Laura Thomson; AddisonWesley Professional; 2009
- JavaScript for Programmers Paul J. Deitel and Harvey M. Deitel; Prentice Hall; 2009