Faculty of Science and Technology

Department of Computer Science

WEB TECHNOLOGIES (CSC 3222)

Lecture Note 8

Week 8

Sazzad Hossain
sazzad@aiub.edu

Contents

PHP MySQL Prepared Statements

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

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

Compared to executing SQL statements directly, prepared statements have three main advantages:

- Prepared statements reduce parsing time as the preparation on the query is done only once (although the statement is executed multiple times)
- Bound parameters minimize bandwidth to the server as you need send only the parameters each time, and not the whole query
- 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.

The following example uses prepared statements and bound parameters in MySQLi:

Example (MySQLi with Prepared Statements)

```
<?php
$servername = "localhost";
$username = "root";//DB default username is root
$password = "";//DB default password in empty/null
$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);

// 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();
?>
```

Code lines to explain from the example above:

"INSERT INTO MyGuests (firstname, lastname, email) VALUES (?, ?, ?)"

In our SQL, we insert a question mark (?) where we want to substitute in an integer, string, double or blob value.

Then, have a look at the 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.

The argument may be one of four types:

- i - integer
- d - double
- s - string

- b - BLOB

We must have one of these for each parameter.

By telling MySQL what type of data to expect, we minimize the risk of SQL injections.

Note: If we want to insert any data from external sources (like user input), it is very important that the data is sanitized and validated.

PHP MySQL Select Data

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

```
SELECT column_name(s) FROM table_name
```

or we can use the * character to select ALL columns from a table:

```
SELECT * FROM table_name
```

The following example selects the id, firstname and lastname columns from the MyGuests table and displays it on the page:

Example (MySQLi Object-oriented)

```
<?php
$servername = "localhost";
$username = "root";//DB default username is root
$password = "";//DB default password in empty/null
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
  die("Connection failed: " . $conn->connect_error);
}

$sql = "SELECT id, firstname, lastname FROM MyGuests";
$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();
?>
```

SQL query that selects the id, firstname and lastname columns from the MyGuests table. The next line of code runs the query and puts the resulting data into a variable called $result.

Then, 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.

PHP MySQL Use The WHERE Clause

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
```

The following example selects the id, firstname and lastname columns from the MyGuests table where the lastname is "Doe", and displays it on the page:

Example (MySQLi Object-oriented)

```
<?php
$servername = "localhost";
$username = "root";//DB default username is root
$password = "";//DB default password in empty/null
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
  die("Connection failed: " . $conn->connect_error);
}

$sql = "SELECT id, firstname, lastname FROM MyGuests WHERE
lastname='Doe'";
$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();
?>
```

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

Then, 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.

PHP MySQL Use The ORDER BY Clause

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
```

The following example selects the id, firstname and lastname columns from the MyGuests table. The records will be ordered by the lastname column.

Example (MySQLi Object-oriented)

```php
<?php
$servername = "localhost";
$username = "root";//DB default username is root
$password = "";//DB default password in empty/null
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
  die("Connection failed: " . $conn->connect_error);
}

$sql = "SELECT id, firstname, lastname FROM MyGuests ORDER BY lastname";
$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();
?>
```

SQL query that selects the id, firstname and lastname columns from the MyGuests 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.

Then, 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.

PHP MySQL Delete Data

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

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

Note: The WHERE clause specifies which record or records that should be deleted. If you omit the WHERE clause, all records will be deleted!

The following examples delete the record with id=3 in the "MyGuests" table:

Example (MySQLi Object-oriented)

```
<?php
$servername = "localhost";
$username = "root";//DB default username is root
$password = "";//DB default password in empty/null
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
  die("Connection failed: " . $conn->connect_error);
}

// sql to delete a record
$sql = "DELETE FROM MyGuests WHERE id=3";

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

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

PHP MySQL 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 specifies which record or records that should be updated. If you omit the WHERE clause, all records will be updated. The following examples update the record with id=2 in the "MyGuests" table:

Example (MySQLi Object-oriented)

```
<?php
$servername = "localhost";
$username = "root";//DB default username is root
$password = "";//DB default password in empty/null
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
  die("Connection failed: " . $conn->connect_error);
}

$sql = "UPDATE MyGuests SET lastname='Doe' WHERE id=2";

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

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

PHP MySQL Limit Data Selections

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". The SQL query would then look like this:

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

When the SQL query above is run, it will return the first 30 records.

What if we want to select records 16 - 25 (inclusive)?

Mysql also provides a way to handle this: by using OFFSET.

The SQL query below "return only 10 records, start on record 16 (OFFSET 15)":

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

You could also use a shorter syntax to achieve the same result:

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

Notice that the numbers are reversed when you use a comma.

PHP Insert File in DB

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)

Inserting file into DB

```
<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();
 }
}
```

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">

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

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\"

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 )
```

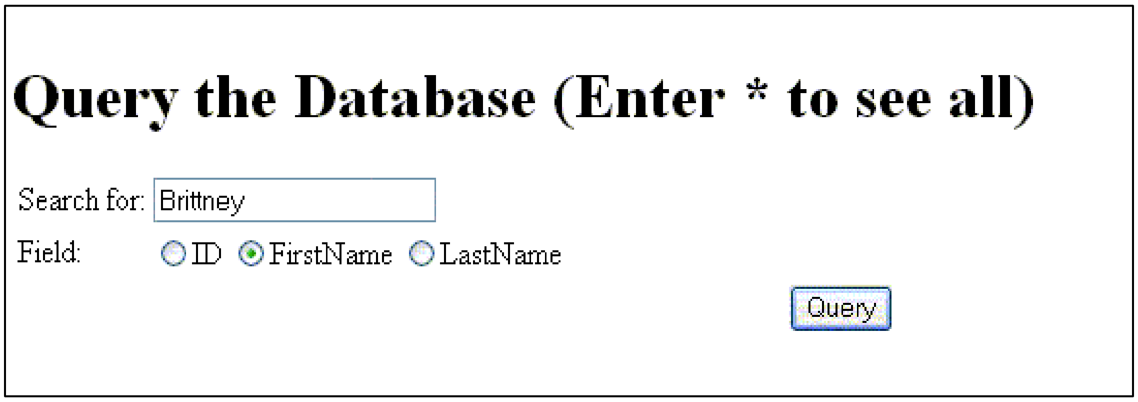
Retrieve/Display Image

```
<?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();
```

```
?>
```

Exercises

1. Create a form that looks similar to the following (values have been typed in for later demonstration)

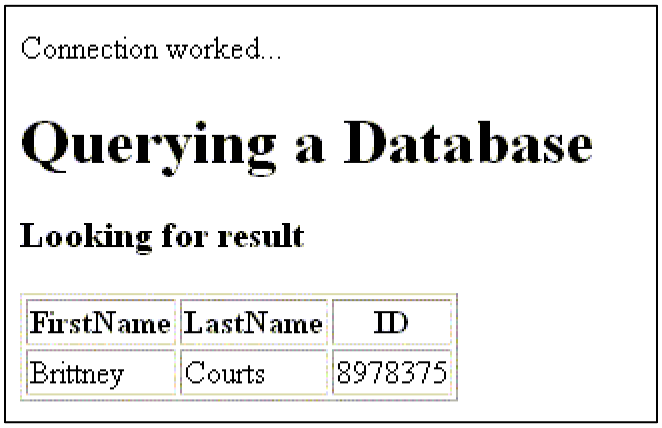


Query the Database (Enter * to see all)

Search for: Brittney

Field: ID [✓ FirstName] LastName

When you click on the "Query" button, below screen will appear



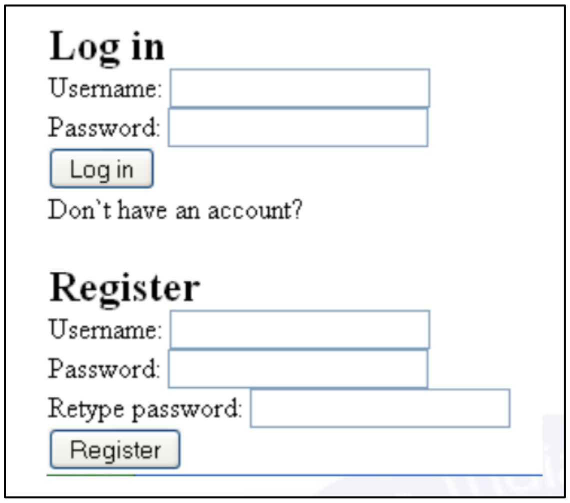
Connection worked...

Querying a Database

Looking for result

FirstName	LastName	ID
Brittney	Courts	8978375

2.



Write a program to register a user and after registration user can use their credentials to login.

3. Write a program to update information of specific user.

4. Write a program to delete information of specific user.

References

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