PHP and MySQL (Advanced)

Introduction

- PHP 5 and later can work with a MySQL database using:
  - **MySQLi extension** (the "i" stands for improved)
  - **PDO (PHP Data Objects)**
- Earlier versions of PHP used the MySQL extension. However, this extension was deprecated in 2012.
- Any of the functions that are prefixed with mysql_ are now being discouraged by PHP themselves.

mysql_query

(PHP 4, PHP 5)

mysql_query — Send a MySQL query

Warning This extension was deprecated in PHP 5.5.0, and it was removed in PHP 7.0.0. Instead, the MySQLi or PDO_MySQL extension should be used. See also MySQL: choosing an API guide and related FAQ for more information. Alternatives to this function include:

- mysqli_query()
- PDO::query()

MySQLi or PDO?

- It would be "Whatever you prefer".
- Both MySQLi and PDO have their advantages:
- PDO will work on 12 different database systems, where as MySQLi will only work with MySQL databases.
- So, if you have to switch your project to use another database, PDO makes the process easy.
  - You only have to change the connection string and a few queries. With MySQLi, you will need to rewrite the entire code - queries included.
- Both are object-oriented, but MySQLi also offers a **procedural API.**
- Both support Prepared Statements. **Prepared Statements protect from SQL injection**, and are very important for web application security.

Working with MySQL Database

- **Create a MySQL Database Using MySQLi and PDO**
- The CREATE DATABASE statement is used to create a database in MySQL.
- E.g. createSQLi.php, createPDO.php
- When you create a new database, you must only specify the first three arguments to the mysqli object (servername, username and password). **Tip:** If you have to use a specific port, add an empty string for the database-name argument, like this: new mysqli("localhost", "username", "password", "", port)
- A great benefit of PDO is that it has exception class to handle any problems that may occur in our database queries.
  - If an exception is thrown within the try{ } block, the script stops executing and flows directly to the first catch(){ } block.

Create MySQL Tables

- The CREATE TABLE statement is used to create a table in MySQL.
- We will create a table named "MyGuests", with five columns: "id", "firstname", "lastname", "email" and "reg_date":
- CREATE TABLE MyGuests (
  id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  firstname VARCHAR(30) NOT NULL,
  lastname VARCHAR(30) NOT NULL,
  email VARCHAR(50),
  reg_date TIMESTAMP
)

Create MySQL Tables

- The data type specifies what type of data the column can hold. For a complete reference of all the available data types, go to our Data Types reference.
- After the data type, you can specify other optional attributes for each column:
- NOT NULL - Each row must contain a value for that column, null values are not allowed
- DEFAULT value - Set a default value that is added when no other value is passed
- UNSIGNED - Used for number types, limits the stored data to positive numbers and zero
- AUTO INCREMENT - MySQL automatically increases the value of the field by 1 each time a new record is added
- PRIMARY KEY - Used to uniquely identify the rows in a table. The column with PRIMARY KEY setting is often an ID number, and is often used with AUTO_INCREMENT

Insert Data Into MySQL

- **Insert Data Into MySQL Using MySQLi and PDO**
- After a database and a table have been created, we can start adding data in them. Here are some syntax rules to follow:
  - The SQL query must be quoted in PHP
  - String values inside the SQL query must be quoted
  - Numeric values must not be quoted
  - The word NULL must not be quoted
- The INSERT INTO statement is used to add new records to a MySQL table:
- INSERT INTO table_name (column1, column2, column3,...) VALUES (value1, value2, value3,...)
- If a column is AUTO_INCREMENT (like the "id" column) or TIMESTAMP (like the "reg_date" column), it is not needed to be specified in the SQL query; MySQL will automatically add the value.
- Multiple SQL statements must be executed with the mysqli_multi_query() function.

Prepared Statements and Bound Parameters

- 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:
  - 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(?, ?, ?)
  - The database parses, compiles, and performs query optimization on the SQL statement template, and stores the result without executing it
  - 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 Statements and Bound Parameters

- Compared to executing SQL statements directly, prepared statements have two main advantages:
  - Prepared statements reduces 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.

Select Data From a MySQL Database

- 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

Delete Data From MySQL

- The DELETE statement is used to delete records from a table:
- DELETE FROM table_name
  WHERE some_column = some_value
- **The WHERE clause in the DELETE syntax:** The WHERE clause specifies which record or records that should be deleted. If you omit the WHERE clause, all records will be deleted!

Update Data In a MySQL

- 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
- **Notice the WHERE clause in the UPDATE syntax:** The WHERE clause specifies which record or records that should be updated. If you omit the WHERE clause, all records will be updated!

Limit Data Selections From MySQL

- 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";

Limit Data Selections From MySQL

- 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 says "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.