# Лабораторная работа №7

Тема: Изучение PHP и взаимодействия с СУБД MySQL.

**Задание**

На основе лабораторной работы №6

Создать базу данных с помощью MySQL, установить соединение между базой и сервером, реализовать добавление постов/статей. Добавить страницу авторизации и регистрации.

**Требования**

* База данных должна содержать две таблицы: учетные записи пользователей и таблица постов/статей;
* Ссылки на страницы авторизации и регистрации должны быть видны в меню навигации;
* Проверять корректность введенных данных и наличие записей в базе данных;

# Справочная информация

База данных – это организованная структура, предназначенная для хранения, изменения и обработки взаимосвязанной информации. MySQL - это популярная система управления базами данных (СУБД), очень часто применяемая в сочетании с PHP.

Соединение с базой данных

MySQLi (от англ. MySQL improved) — расширение драйвера реляционных баз данных, используемого в языке программирования PHP для предоставления доступа к базам данных MySQL. MySQLi является обновлённой версией драйвера PHP MySQL, и даёт различные улучшения в работе с базами данных. Основным преимуществом MySQLi является то, что при помощи предварительно объявляемых операторов, могут быть предотвращены так называемые SQL-инъекции.

MySqli предлагает два способа соединения с базой данных: процедурный и объектно-ориентированный. Рекомендуется использовать объектно-ориентированный. PHP

//процедурный стиль

$mysqli = mysqli\_connect('host','username','password','database\_name');

//объектно-ориентированный стиль (рекомендуется)

$mysqli = new mysqli('host','username','password','database\_name');

Ниже показано открытие соединения с базой данных объектно-ориентированным способом. Этот способ будет использоваться и во всех приведенных ниже примерах.

<?php

// Открываем новое соединение с MySQL сервером

$mysqli = new mysqli('host','username','password','database\_name');

//Выводим любую ошибку соединения

if ($mysqli->connect\_error) {

die('Error : ('. $mysqli->connect\_errno .') '. $mysqli->connect\_error);

}

?>

## Выбор (SELECT) результирующего ряда в виде ассоциативного массива

mysqli\_fetch\_assoc() : в приведенном ниже коде происходит извлечение результирующего ряда в виде ассоциативного массива. Возвращаемый массив содержит строки, полученные из базы данных, где имена столбцов будут являться ключом, используемым для доступа к внутренним данным. Как показана ниже, данные отображаются в виде HTML таблицы.

<?php

//Open a new connection to the MySQL server

$mysqli = new mysqli('host','username','password','database\_name');

//Output any connection error

if ($mysqli->connect\_error) {

die('Error : ('. $mysqli->connect\_errno .') '. $mysqli->connect\_error);

}

//MySqli Select Query

$results = $mysqli->query("SELECT id, product\_code, product\_desc, price FROM products");

print '<table border="1">';

while($row = $results->fetch\_assoc()) {

print '<tr>';

print '<td>'.$row["id"].'</td>';

print '<td>'.$row["product\_code"].'</td>';

print '<td>'.$row["product\_name"].'</td>';

print '<td>'.$row["product\_desc"].'</td>';

print '<td>'.$row["price"].'</td>';

print '</tr>';

}

print '</table>';

// Frees the memory associated with a result

$results->free();

// close connection

$mysqli->close();

?>

## Выбор (SELECT) результирующего ряда в виде массива (ассоциативный, обычный, или в оба)

Фукнция fetch\_array() : возвращает массив с объединенным функционалом mysqli\_fetch\_row и mysqli\_fetch assoc. Эта функция является расширенной версией функции mysqli\_fetch\_row(); для доступа к данным можно использовать как строку, так и числа.

<?php

//Open a new connection to the MySQL server

$mysqli = new mysqli('host','username','password','database\_name');

//Output any connection error

if ($mysqli->connect\_error) {

die('Error : ('. $mysqli->connect\_errno .') '. $mysqli->connect\_error);

}

//MySqli Select Query

$results = $mysqli->query("SELECT id, product\_code, product\_desc, price FROM products");

print '<table border="1"';

while($row = $results->fetch\_array()) {

print '<tr>';

print '<td>'.$row["id"].'</td>';

print '<td>'.$row["product\_code"].'</td>';

print '<td>'.$row["product\_name"].'</td>';

print '<td>'.$row["product\_desc"].'</td>';

print '<td>'.$row["price"].'</td>';

print '</tr>';

}

print '</table>';

// Frees the memory associated with a result

$results->free();

// close connection

$mysqli->close();

?>

## Выбор (SELECT) результирующего ряда в виде объекта

fetch\_object() : чтобы получить результирующий набор в виде объекта, нужно воспользоваться MySqli fetch\_object(). Атрибуты объекта будут отображать имена полей, найденных внутри результирующего набора.

<?php

//Open a new connection to the MySQL server

$mysqli = new mysqli('host','username','password','database\_name');

//Output any connection error

if ($mysqli->connect\_error) {

die('Error : ('. $mysqli->connect\_errno .') '. $mysqli->connect\_error);

}

//MySqli Select Query

$results = $mysqli->query("SELECT id, product\_code, product\_desc, price FROM products");

print '<table border="1">';

while($row = $results->fetch\_object()) {

print '<tr>';

print '<td>'.$row->id.'</td>';

print '<td>'.$row->product\_code.'</td>';

print '<td>'.$row->product\_name.'</td>';

print '<td>'.$row->product\_desc.'</td>';

print '<td>'.$row->price.'</td>';

print '</tr>';

}

print '</table>';

// close connection

$mysqli->close();

?>

## Выбор (SELECT) одиночного значение

Одиночное значение получить из базы данных можно посредством fetch\_object (метод Cameron Spear).

<?php

//Open a new connection to the MySQL server

$mysqli = new mysqli('host','username','password','database\_name');

//Output any connection error

if ($mysqli->connect\_error) {

die('Error : ('. $mysqli->connect\_errno .') '. $mysqli->connect\_error);

}

//chained PHP functions

$product\_name = $mysqli->query("SELECT product\_name FROM products WHERE id = 1")->fetch\_object()->product\_name;

print $product\_name; //output value

$mysqli->close();

?>

## Извлекаем (SELECT COUNT) количество строк в таблице

Иногда нужно узнать количество строк в таблице, особенно при нумерации страниц.

<?php

//Open a new connection to the MySQL server

$mysqli = new mysqli('host','username','password','database\_name');

//Output any connection error

if ($mysqli->connect\_error) {

die('Error : ('. $mysqli->connect\_errno .') '. $mysqli->connect\_error);

}

//get total number of records

$results = $mysqli->query("SELECT COUNT(\*) FROM users");

$get\_total\_rows = $results->fetch\_row(); //hold total records in variable

$mysqli->close();

?>

## Выбор (SELECT) с помощью шаблонов (prepared statements)

prepared statements — специальный инструмент СУБД, позволяющий ускорить последовательное выполнение повторяющихся запросов, построенных по одному и тому же шаблону.

Одной из особенностей MySqli является возможность использования уже написанных шаблонов: то есть запрос достаточно написать один раз, после чего его можно многократно исполнять с различными параметрами. Использование уже написанных шаблонов улучшает производительность для больших таблицах и сложных запросов. Для предотвращения попадания вредоносного кода анализ каждого запроса производится сервером отдельно.

Код ниже использует шаблон (Prepared statement), чтобы получать данные из базы данных. Заполнитель ? в запросе SQL играет роль маркера и будет замещен параметром, который, в свою очередь, может быть строкой, целым числом, double или blob. В нашем случае это строка $search\_product.

PHP

$search\_product = "PD1001"; //product id

//create a prepared statement

$query = "SELECT id, product\_code, product\_desc, price FROM products WHERE product\_code=?";

$statement = $mysqli->prepare($query);

//bind parameters for markers, where (s = string, i = integer, d = double, b = blob)

$statement->bind\_param('s', $search\_product);

//execute query

$statement->execute();

//bind result variables

$statement->bind\_result($id, $product\_code, $product\_desc, $price);

print '<table border="1">';

//fetch records

while($statement->fetch()) {

print '<tr>';

print '<td>'.$id.'</td>';

print '<td>'.$product\_code.'</td>';

print '<td>'.$product\_desc.'</td>';

print '<td>'.$price.'</td>';

print '</tr>';

}

print '</table>';

//close connection

$statement->close();

Тот же запрос с несколькими параметрами:

PHP

$search\_ID = 1;

$search\_product = "PD1001";

$query = "SELECT id, product\_code, product\_desc, price FROM products WHERE ID=? AND product\_code=?";

$statement = $mysqli->prepare($query);

$statement->bind\_param('is', $search\_ID, $search\_product);

$statement->execute();

$statement->bind\_result($id, $product\_code, $product\_desc, $price);

print '<table border="1">';

while($statement->fetch()) {

print '<tr>';

print '<td>'.$id.'</td>';

print '<td>'.$product\_code.'</td>';

print '<td>'.$product\_desc.'</td>';

print '<td>'.$price.'</td>';

print '</tr>';

}

print '</table>';

//close connection

$statement->close();

## Вставка (INSERT) записи

Запись ниже вставляет в таблицу новый ряд.

PHP

<?php

//values to be inserted in database table

$product\_code = '"'.$mysqli->real\_escape\_string('P1234').'"';

$product\_name = '"'.$mysqli->real\_escape\_string('42 inch TV').'"';

$product\_price = '"'.$mysqli->real\_escape\_string('600').'"';

//MySqli Insert Query

$insert\_row = $mysqli->query("INSERT INTO products (product\_code, product\_name, price) VALUES($product\_code, $product\_name, $product\_price)");

if($insert\_row){

print 'Success! ID of last inserted record is : ' .$mysqli->insert\_id .'<br />';

}else{

die('Error : ('. $mysqli->errno .') '. $mysqli->error);

}

?>

Отрывок ниже вставляет те же значения посредством шаблонов (Prepared Statement). Как мы уже говорили, шаблоны чрезвычайно эффективны против SQL инъекция. Для приведенного примера их использование является оптимальным вариантом.

PHP

//values to be inserted in database table

$product\_code = 'P1234';

$product\_name = '42 inch TV';

$product\_price = '600';

$query = "INSERT INTO products (product\_code, product\_name, price) VALUES(?, ?, ?)";

$statement = $mysqli->prepare($query);

//bind parameters for markers, where (s = string, i = integer, d = double, b = blob)

$statement->bind\_param('sss', $product\_code, $product\_name, $product\_price);

if($statement->execute()){

print 'Success! ID of last inserted record is : ' .$statement->insert\_id .'<br />';

}else{

die('Error : ('. $mysqli->errno .') '. $mysqli->error);

}

$statement->close();

## Вставка (INSERT) нескольких записей

Вставка нескольких рядов одновременно осуществляется путем включения ряда значений столбцов, где каждый ряд значений должен быть обнесен скобками и отделен от других запятой. Иногда нужно узнать, сколько записей было вставлено, обновлено или удалено, для этого можно воспользоваться mysqli\_affected\_rows.

PHP

//product 1

$product\_code1 = '"'.$mysqli->real\_escape\_string('P1').'"';

$product\_name1 = '"'.$mysqli->real\_escape\_string('Google Nexus').'"';

$product\_price1 = '"'.$mysqli->real\_escape\_string('149').'"';

//product 2

$product\_code2 = '"'.$mysqli->real\_escape\_string('P2').'"';

$product\_name2 = '"'.$mysqli->real\_escape\_string('Apple iPad 2').'"';

$product\_price2 = '"'.$mysqli->real\_escape\_string('217').'"';

//product 3

$product\_code3 = '"'.$mysqli->real\_escape\_string('P3').'"';

$product\_name3 = '"'.$mysqli->real\_escape\_string('Samsung Galaxy Note').'"';

$product\_price3 = '"'.$mysqli->real\_escape\_string('259').'"';

//Insert multiple rows

$insert = $mysqli->query("INSERT INTO products(product\_code, product\_name, price) VALUES

($product\_code1, $product\_name1, $product\_price1),

($product\_code2, $product\_name2, $product\_price2),

($product\_code3, $product\_name3, $product\_price3)");

if($insert){

//return total inserted records using mysqli\_affected\_rows

print 'Success! Total ' .$mysqli->affected\_rows .' rows added.<br />';

}else{

die('Error : ('. $mysqli->errno .') '. $mysqli->error);

}

## Обновление (Update)/удаление (Delete) записей

Принцип обновление и удаление записей тот же. Достаточно заменить строку запроса на MySql update или delete (не понял, сам смотри).

PHP

//MySqli Update Query

$results = $mysqli->query("UPDATE products SET product\_name='52 inch TV', product\_code='323343' WHERE ID=24");

//MySqli Delete Query

//$results = $mysqli->query("DELETE FROM products WHERE ID=24");

if($results){

print 'Success! record updated / deleted';

}else{

print 'Error : ('. $mysqli->errno .') '. $mysqli->error;

}

## Обновление с помощью шаблонов (prepared statements)

Пример обновления записи с помощью шаблонов (prepared statements) приведен ниже.

PHP

$product\_name = '52 inch TV';

$product\_code = '9879798';

$find\_id = 24;

$query = "UPDATE products SET product\_name=?, product\_code=? WHERE ID=?";

$statement = $mysqli->prepare($query);

//bind parameters for markers, where (s = string, i = integer, d = double, b = blob)

$results = $statement->bind\_param('ssi', $product\_name, $product\_code, $find\_id);

if($results){

print 'Success! record updated';

}else{

print 'Error : ('. $mysqli->errno .') '. $mysqli->error;

}

## Удаление старых записей

Удалению подвергаются все записи, находящиеся на сервере больше 1 дня; количество дней можно задать самому.

PHP

//MySqli Delete Query

$results = $mysqli-<query("DELETE FROM products WHERE added\_timestamp > (NOW() - INTERVAL 1 DAY)");

if($results){

print 'Success! deleted one day old records';

}else{

print 'Error : ('. $mysqli-<errno .') '. $mysqli-<error;

}