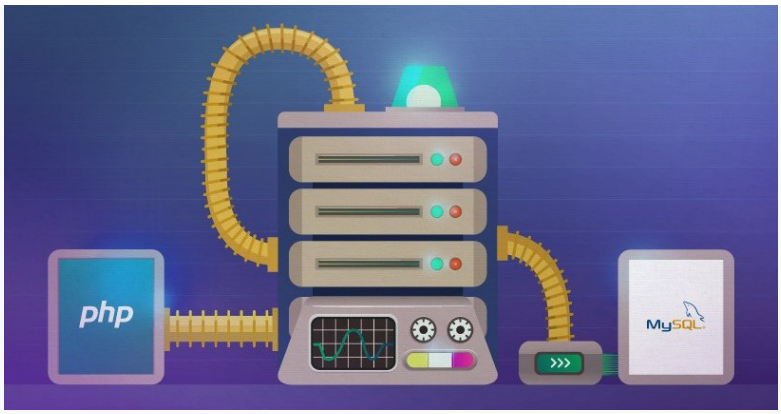
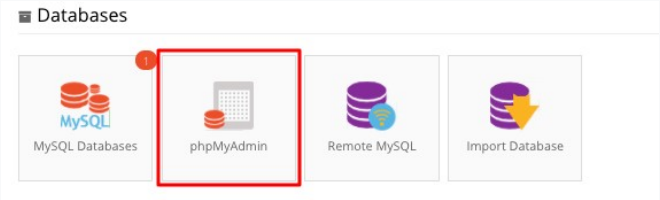
Як використовувати PHP для вставки рядків в базу даних MySQL

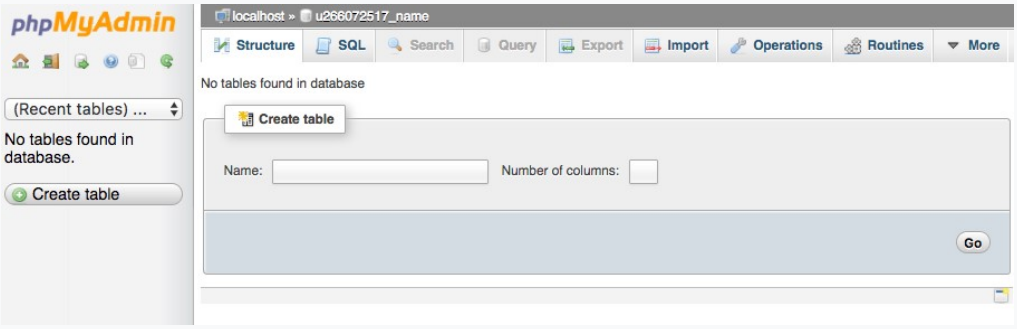


Крок 1 - Створення таблиці

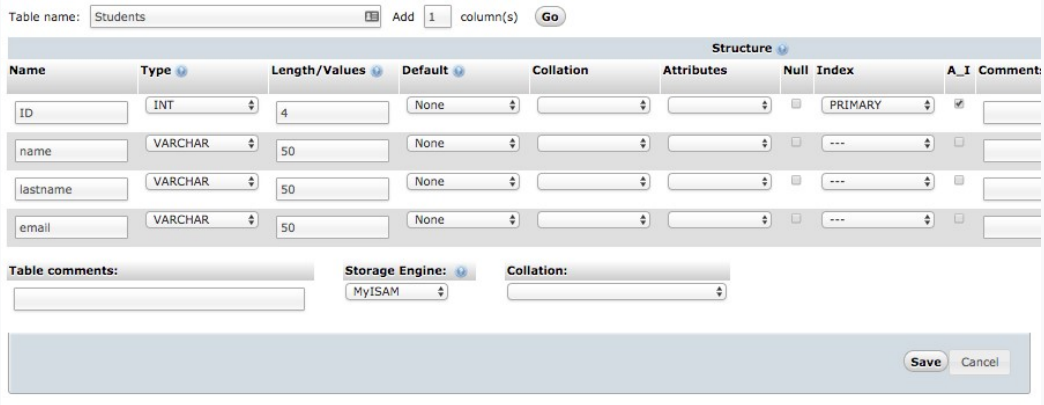
Спочатку потрібно створити таблицю для даних. Це проста процедура, яку можна виконати за допомогою **phpMyAdmin** в панелі управління хостингом.



Після входу ви phpMyAdmin ви побачите такий інтерфейс:



Створимо в базі даних u266072517_name таблицю з ім'ям Students, натиснувши на кнопку «Створити таблицю». Після цього ви побачите нову сторінку, на якій задаються всі необхідні параметри таблиці:



Це найпростіше налаштування, яке можна використовувати для таблиці та отримання додаткової інформації про структуру таблиць / баз даних.

Параметри стовпців:

- **Name** - це ім'я стовпця, яке відображається у верхній частині таблиці.
- **Type** - тип стовпчика. Наприклад, ми вибрали varchar, тому що будемо вводити строкові значення.
- **Length / Values** - використовується для вказівки максимальної довжини, яку може мати запис в цьому стовпці.
- **Index** - ми використовували «Первинний» індекс для поля «ID». При створенні таблиці рекомендується застосовувати в якості первинного ключа тільки один стовпець. Він використовується для перерахування записів в таблиці і потрібно при налаштуванні таблиці. Я також відзначив «A_I», що означає «Auto Increment» - параметр автоматичного присвоєння номера записів (1,2,3,4 ...).

Натисніть кнопку «Зберегти», і таблиця буде створена.

Крок 2. Написання PHP-коду для вставки даних в MySQL.

Варіант 1 - метод MySQLi

Спочатку необхідно встановити з'єднання з базою даних. Після цього використовуємо SQL-запит INSERT. Повний приклад коду:

```php
<?php
    $servername = "mysql.hostinger.co.uk";
    $database = "u266072517_name";
    $username = "u266072517_user";
    $password = "buystuffpwd";

    // Устанавливаем соединение

    $conn = mysqli_connect($servername, $username, $password, $database);

    // Проверяем соединение

    if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
    }

    echo "Connectedsuccessfully";

    $sql = "INSERT INTO Students (name, lastname, email) VALUES ('Thom', 'Vial', 'thom.v@some.com')";
    if (mysqli_query($conn, $sql)) {
          echo "New recordcreatedsuccessfully";
    } else {
          echo "Error: " . $sql . "<br>" . mysqli_error($conn);
    }
    mysqli_close($conn);

    ?>

```

Перша частина коду (рядки 3 - 18) призначена для підключення до бази даних. Почнемо з рядка № 19:

```php
$sql = "INSERT INTO Students (name, lastname, email) VALUES ('Thom', 'Vial', 'thom.v@some.com')";
```

Він вставляє дані в базу MySQL. INSERT INTO - це оператор, який додає дані в зазначену таблицю. У цьому прикладі дані додаються до таблиці Students.

Далі перераховуються стовпці, в які вставляються значення: name, lastname, email. Будьте уважними - дані будуть додані в зазначеному порядку. Якби ми написали (email, lastname, name), значення б були додані в іншому порядку.

Наступна частина - це оператор VALUES. Тут ми вказуємо значення для стовпців: name = Thom, lastname = Vial, email = thom.v@some.com.

Ми запустили запит з використанням PHP-коду. У програмному коді SQL-запити повинні бути екрановані лапками. Наступна частина коду (рядки 20-22) перевіряє, чи був наш запит успішним:

```
1  if (mysqli_query($conn, $sql)) {
2          echo "New recordcreatedsuccessfully";
3      }
```

Цей код виводить повідомлення про успішне виконання запиту.

І остання частина (рядки 22 - 24) відображає повідомлення, якщо запит не був успішним:

```
1  else {
2          echo "Error: " . $sql . "<br>" . mysqli_error($conn);
3      }
```

Варіант 2 - метод об'єкта даних PHP (PDO)

Спочатку нам потрібно підключитися до бази даних шляхом створення нового об'єкта PDO. При роботі з ним будемо використовувати різні методи PDO. Методи об'єктів викликаються наступним чином:

```
1  $the_Object->the_Method();
```

PDO дозволяє «підготувати» SQL-код до його виконання. SQL-запит оцінюється і «виправляється» перед запуском. Наприклад, найпростіша атака з використанням SQL-ін'єкції може бути виконана через просте введення SQL-коду в поле форми. наприклад:

```
1  // Пользователь пишет это в поле имени пользователя формы авторизации
2  john"; DROP DATABASE user_table;
3
4  // Окончательный запрос будет следующим
5  "SELECT * FROM user_table WHERE username = john"; DROP DATABASE user_table;
```

Так як це синтаксично правильний SQL- код, крапка з комою робить DROP DATABASE user_table новим SQL-запитом, і призначена для користувача таблиця видаляється. Підготовлені вирази (*зв'язані змінні*) не дозволяють, щоб крапка з комою і лапки завершували вихідний запит. Тому команда DROP DATABASE ніколи не буде виконана.

Щоб використовувати підготовлені вирази, потрібно написати нову змінну, яка викликає метод **prepare ()** об'єкта бази даних.

Коректний код:

```
1|  <?php
2|  $servername = "mysql.hostinger.com";
3|  $database = "u266072517_name";
4|  $username = "u266072517_user";
5|  $password = "buystuffpwd";
6|  $sql = "mysql:host=$servername;dbname=$database;";
7|  $dsn_Options = [PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION];
8|
```

```
9|  // Создаем новое соединение с базой данных MySQL, используя PDO, $my Db Connection - это объект
10| try {
11| $my Db Connection = new PDO($sql, $username, $password, $dsn Options);
12| echo "Connectedsuccessfully";
13| } catch (PDOException $error) {
14| echo 'Connection error: ' . $error->getMessage();
15| }
16|
17| // Устанавливаем переменные для персоны, которую мы хотим добавить в базу данных
18| $first_Name = "Thom";
19| $last Name = "Vial";
20| $email = "thom.v@some.com";
21|
22| // Создаем переменную, которая вызывает методобъекта базы данных prepare()
23| // Запрос SQL, который вы хотите выполнить, вводится как параметр, а заполнители пишутся
    следующим образом:placeholder_name
24|
25| $my_Insert_Statement = $my_Db_Connection->prepare("INSERT INTO Students (name, lastname, email)
    VALUES (:first name, :last name, :email)");
26|
27| // Теперь мы указываем скрипту, какая переменная ссылается на каждый заполнитель, чтобы
    использовать метод bindParam()
28| // Первый параметр - это заполнитель в операторе выше, второй - это переменная, на которую он
    должен ссылаться
29|
30| $my_Insert_Statement->bindParam(:first_name, $first_Name);
31| $my Insert Statement->bindParam(:last name, $last Name);
32| $my_Insert_Statement->bindParam(:email, $email);
33|
34| // Выполняем запрос, используя данные, которые только что определили
35| // Метод execute() возвращает TRUE, если он выполнен успешно, и FALSE, если нет, предоставляя
    вам возможность вывести собственное сообщение
36|
37| if ($my_Insert_Statement->execute()) {
38| echo "New recordcreatedsuccessfully";
39| } else {
40| echo "Unable to createrecord";
41| }
42|
43| // В этой точке можно изменить данные переменных и выполнить запрос, чтобы добавить другие
    данные в базу
44|
45| data to the database
46| $first_Name = "John";
47| $last_Name = "Smith";
48| $email = "john.smith@email.com";
49| $my_Insert_Statement->execute();
50|
51| // Выполняем снова, когда переменная изменена
52|
53| if ($my_Insert_Statement->execute()) {
54| echo "New recordcreatedsuccessfully";
55| } else {
56| echo "Unable to createrecord";
57|
```

У рядках 28, 29 і 30 ми використовуємо метод **bindParam ()** об'єкта бази даних. Також існує метод **bindValue ()**, який сильно відрізняється від попереднього.

- **bindParam ()** – цей метод оцінює дані при досягненні методу execute (). У перший раз, коли скрипт досягає методу execute (), він бачить, що $ first_Name відповідає «Thom». Потім пов'язує це значення і запускає запит. Коли скрипт досягає другого методу execute (), він бачить, що $ first_Name тепер відповідає «John». Після чого пов'язує це значення і знову запускає

запит до нових значень. Важливо пам'ятати, що ми одного разу визначили запит і повторно використовуємо його з різними даними в різних точках скрипта.

- **bindValue ()** – цей метод оцінює дані, як тільки досягається bindValue (). Оскільки для $ first_Name було встановлено значення «Thom», при досягненні bindValue (), воно буде використовуватися кожен раз, коли викликається метод execute () для $ my_Insert_Statement.

Зверніть увагу, що ми повторно використовуємо змінну $ first_Name і присвоюємо її нове значення вдруге. Після запуску скрипта в БД будуть вказані обидва імені, не зважаючи на те, що змінна $ first_Name в кінці скрипта має значення «John». Пам'ятайте, що PHP перевіряє весь скрипт, перш ніж запустити його.

Якщо ви відновите скрипт, щоб замінити bindParam на bindValue, ви двічі вставите в базу даних «Thom Vial», а John Smith буде проігнорований.

Крок 3 - підтвердження успішного виконання і вирішення проблем

Якщо запит на вставку рядків в базу не був успішним, ми побачимо наступне повідомлення:

```
Connected successfully
Error: INSERT INTO Students {name, lastname, email} VALUES ('Thom', 'Vial', 'thom.v@some.com')
You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '{name, lastname, email} VALUES ('Thom', 'Vial', 'thom.v@some.com')' at line 1
```

(Щоб побачите це повідомлення в коді зробимо одну синтаксичну помилку)

Перша частина коду в порядку, з'єднання було успішно встановлено, але SQL-запит не пройшов.

```
1|  "Error: INSERT INTO Students {name, lastname, email} VALUES ('Thom', 'Vial', 'thom.v@some.com')
2|  You have an error in your SQL syntax; check the manualthatcorresponds to your MySQL server
    version for the rightsyntax to use near '{name, lastname, email} VALUES ('Thom', 'Vial',
    'thom.v@some.com')' at line 1
```

Була допущена синтаксична помилка, яка викликала збій скрипта. Помилка була тут:

```
1|  $sql = "INSERT INTO Students {name, lastname, email} VALUES ('Thom', 'Vial',
    'thom.v@some.com')";
```

Ми використовували фігурні дужки замість звичайних. Це невірно, і скрипт видав синтаксичну помилку.

PDO

У рядку 7 з'єднання PDO для режиму помилок встановлено «display all exceptions». Якщо задано інше значення, і запит не вдався б, ми не змогли б отримати ніяких повідомлень про помилки.

Цю установку слід використовувати тільки при розробці скрипта. При її активації можуть відображатися імена бази даних і таблиць, які краще приховати

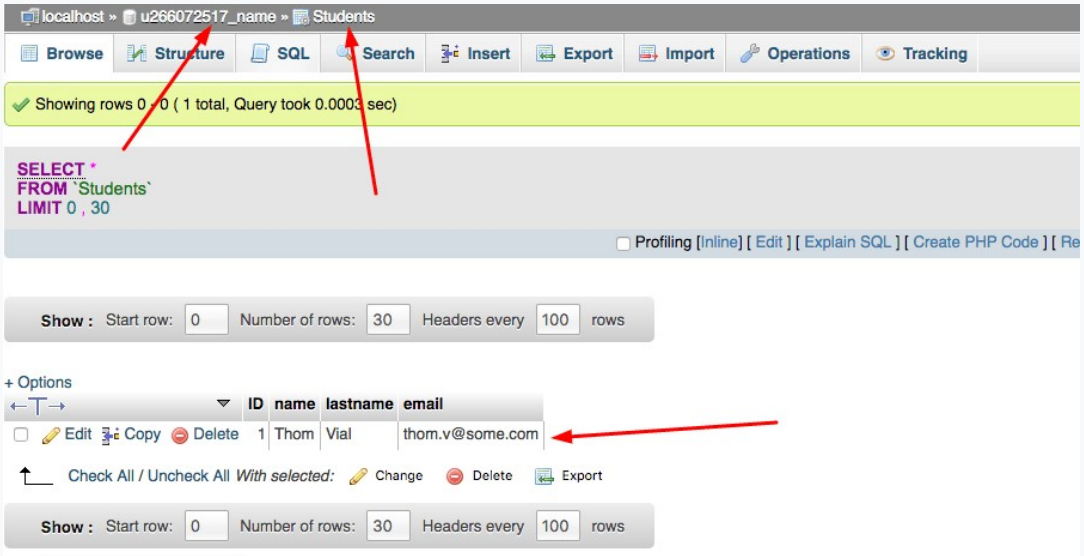
з метою безпеки. У наведеному вище випадку, коли замість звичайних дужок використовувалися фігурні, повідомлення про помилку виглядає так:

```
1|  Fatal error: Uncaughtexception 'PDOException' with message 'SQLSTATE[42000]: Syntax error or
    accessviolation: 1064 You have an error in your SQL syntax; <code>check the
    manualthatcorresponds to your MySQL server version for the rightsyntax to use near '{name,
    lastname, email} VALUES ('Thom', 'Vial', 'thom.v@some.com')' at line 1"</code>
```

Інші можливі проблеми:

- Невірно вказані стовпці (неіснуючі стовпці або орфографічна помилка в їх іменах).
- Один тип значення присвоюється колонки іншого типу. Наприклад, якщо спробувати вставити число 47 в стовпець Name, то отримаємо помилку. У цьому стовпці необхідно використовувати строкове значення. Але якби ми вказали число в лапках (наприклад, «47») то спрацювало б, тому що це рядок.
- Спроба ввести дані в таблицю, яка не існує. А також допущена орфографічна помилка в імені таблиці.

Після успішного введення даних ми побачимо, що вони додані в базу даних. Нижче наведено приклад таблиці, в яку додали дані.



Використані матеріали з сайту https://www.internet-technologies.ru/