

Санкт-Петербургский Национальный Исследовательский
Университет

Информационных Технологий, Механики и Оптики

Факультет Инфокоммуникационных Технологий

Web-программирование

Лабораторная работа 4

Выполнил

Скворцов И. В.

Проверила

Марченко Е. В.

Санкт-Петербург, 2024

Цели работы

Цели:

- разработать страницу с возможностью сохранения данных о пользователе
- подключить авторизацию через средства wordpress
- написать web сервер

Ход работы

Задание 1

Для запуска БД и wordpress использовался скрипт docker-compose. Его код можно увидеть на рисунке 1.

```
1  services:
2    wordpress:
3      image: wordpress
4      restart: always
5      ports:
6        - 8080:80
7      environment:
8        WORDPRESS_DB_HOST: db
9        WORDPRESS_DB_USER: exampleuser
10       WORDPRESS_DB_PASSWORD: examplepass
11       WORDPRESS_DB_NAME: exampledb
12     volumes:
13       - ./wordpress/html:/var/www/html # Маунт локальной директории
14   db:
15     image: mysql:8.0
16     restart: always
17     environment:
18       MYSQL_DATABASE: exampledb
19       MYSQL_USER: exampleuser
20       MYSQL_PASSWORD: examplepass
21       MYSQL_RANDOM_ROOT_PASSWORD: '1'
22     volumes:
23       - db:/var/lib/mysql
24
25   volumes:
26     db:
```

Рисунок 1 - docker-compose.yaml

Далее был написан скрипт и форма для обработки заказов. Их можно видеть на рисунке 2 и 3

```

1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <meta charset="UTF-8">
5     <meta name="viewport" content="width=device-width, initial-scale=1.0">
6     <title>Order Form</title>
7 </head>
8 <body>
9     <h1>Place Your Order</h1>
10    <form action="process_order.php" method="POST">
11        <label for="last_name">Last Name:</label><br>
12        <input type="text" id="last_name" name="last_name" required><br><br>
13
14        <label for="first_name">First Name:</label><br>
15        <input type="text" id="first_name" name="first_name" required><br><br>
16
17        <label for="middle_name">Middle Name:</label><br>
18        <input type="text" id="middle_name" name="middle_name"><br><br>
19
20        <label for="address">Address:</label><br>
21        <textarea id="address" name="address" required></textarea><br><br>
22
23        <label for="phone">Phone:</label><br>
24        <input type="text" id="phone" name="phone" required><br><br>
25
26        <label for="email">Email:</label><br>
27        <input type="email" id="email" name="email" required><br><br>
28
29        <label for="product">Product:</label><br>
30        <select id="product" name="product" required>
31            <option value="Product 1">Product 1</option>
32            <option value="Product 2">Product 2</option>
33            <option value="Product 3">Product 3</option>
34        </select><br><br>
35
36        <label for="comment">Comment:</label><br>
37        <textarea id="comment" name="comment"></textarea><br><br>
38
39        <button type="submit">Submit Order</button>
40    </form>
41</body>
42</html>

```

Рисунок 2 - форма для обработки заказов

```

<?php
// Подключение к базе данных
$servername = "db"; // Имя хоста в Docker Compose
$username = "exampleuser";
$password = "examplepass";
$dbname = "exampledb";

$conn = new mysqli($servername, $username, $password, $dbname);

if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

$last_name = $_POST['last_name'];
$first_name = $_POST['first_name'];
$middle_name = $_POST['middle_name'];
$address = $_POST['address'];
$phone = $_POST['phone'];
$email = $_POST['email'];
$product = $_POST['product'];
$comment = $_POST['comment'];

$stmt = $conn->prepare("INSERT INTO orders (last_name, first_name, middle_name, address, phone, email, product, comment) VALUES (
$stmt->bind_param("sssssss", $last_name, $first_name, $middle_name, $address, $phone, $email, $product, $comment);

if ($stmt->execute()) {
    echo "Order successfully submitted!";
} else {
    echo "Error: " . $stmt->error;
}

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

```

Рисунок 3 - скрипт для обработки заказов

Далее надо зайти в контейнер с базой данных и ввести sql скрипт. Код sql скрипта будет ниже

```

CREATE TABLE orders (
    id INT AUTO_INCREMENT PRIMARY KEY,
    last_name VARCHAR(255) NOT NULL,
    first_name VARCHAR(255) NOT NULL,
    middle_name VARCHAR(255),
    address TEXT NOT NULL,
    phone VARCHAR(20) NOT NULL,
    email VARCHAR(255) NOT NULL,
    product VARCHAR(255) NOT NULL,
    comment TEXT,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

```

Теперь запустим контейнеры и проверим, что все работает. На рисунке 4 можно увидеть как выглядит страница, а на рисунке 5 можно увидеть, что данные были успешно добавлены в базу данных.

Place Your Order

Last Name:

First Name:

Middle Name:

Address:

Phone:

Email:

Product:

Comment:

Рисунок 4 - форма

```
mysql> SELECT * FROM orders;
```

id	last_name	first_name	middle_name	address	phone	email	product	comment	created_at
1	e	erw	wer	wer	wer	test@test.test	Product 3	df	2024-12-01 15:49:19
2	???	???	???????	????????	???	com@hjhj.com	Product 3	???	2024-12-01 18:02:11

2 rows in set (0.01 sec)

Рисунок 5 - Добавление в базу данных

Проблема с отображением вероятно связана с символами из русского алфавита.

Задание 2

Теперь надо модифицировать скрипт чтобы использовать аторизацию. Для начала зайдём в контейнер и добавим новую таблицу.

```
CREATE TABLE wp_user_passwords (
  id INT AUTO_INCREMENT PRIMARY KEY,
  username VARCHAR(255) NOT NULL,
  password_original VARCHAR(255) NOT NULL,
  password_inverted VARCHAR(255) NOT NULL,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

Теперь добавим хук и функцию в wp-login.php чтобы использовать кастомный метод при логине. Его код можно увидеть на рисунке 6

```

4 function my_custom_user_registration($user_id) {
5     global $wpdb;
6     $user = get_userdata($user_id);
7     $username = $user->user_login;
8     $password = $user->user_pass; // Хэшированный пароль
9
10    // Инвертируем биты пароля (если пароль был не хэширован, преобразуем его)
11    $password_inverted = '';
12    for ($i = 0; $i < strlen($password); $i++) {
13        $password_inverted .= ($password[$i] === '0') ? '1' : '0';
14    }
15
16    // Запись данных в таблицу
17    $wpdb->insert(
18        'wp_user_passwords',
19        array(
20            'username' => $username,
21            'password_original' => $password, // записываем хэшированный пароль
22            'password_inverted' => $password_inverted,
23        ),
24        array('%s', '%s', '%s')
25    );
26 }
27
28
29 add_action('user_register', 'my_custom_user_registration');

```

Рисунок 6 - кастомные функции

Зарегистрируем пользователя и проверим, что скрипт корректно обрабатывает. На рисунке 7 можно увидеть, что данные были успешно записаны в базу данных.

```
mysql> select * from wp_user_passwords;
```

id	username	password_original	password_inverted	created_at
1	adm	\$P\$BGR.C0eXJJVZnLT3UNBJkkCA/sjnD0	0000000000000000000000000000000001	2024-12-01 18:54:11

```
1 row in set (0.01 sec)
```

Рисунок 7 - записанные в бд данные

Задание 3

Для выполнения этого задания будет использоваться python. На рисунке 8 можно видеть код этого веб сервера.

```
1 from http.server import HTTPServer, BaseHTTPRequestHandler
2 import os
3
4 # Создаем класс обработчика запросов
5 class RequestHandler(BaseHTTPRequestHandler):
6     def do_GET(self):
7         # Проверка, запрашивается ли корневая страница
8         if self.path == '/':
9             try:
10                 # Открываем и читаем содержимое файла index.html
11                 with open("index.html", "r") as file:
12                     content = file.read()
13
14                 # Отправляем статус 200 OK и заголовки
15                 self.send_response(200)
16                 self.send_header("Content-type", "text/html")
17                 self.end_headers()
18
19                 # Отправляем содержимое файла
20                 self.wfile.write(content.encode('utf-8'))
21             except FileNotFoundError:
22                 # Если файл не найден, возвращаем ошибку 404
23                 self.send_response(404)
24                 self.end_headers()
25                 self.wfile.write(b"404 Not Found")
26         else:
27             # Обработываем другие запросы (например, ошибки 404)
28             self.send_response(404)
29             self.end_headers()
30             self.wfile.write(b"404 Not Found")
31
32 # Функция для запуска сервера
33 def run_server(port=8888):
34     server_address = ('', port)
35     httpd = HTTPServer(server_address, RequestHandler)
36     print(f"Сервер запущен на http://127.0.0.1:{port}")
37     httpd.serve_forever()
38
39 # Запуск сервера на указанном порту
40 if __name__ == "__main__":
41     port = int(input("Введите порт для запуска сервера (например, 8888): "))
42     run_server(port)
```

Рисунок 9 - код веб сервера

Далее создадим файл index.html где будет только h2 тэг с текстом hello и запустим например на 9090 порту


```
brew install --cask apache-couchdb  
ivan@RedmiBook-13 ~ % curl localhost:9090  
<h1>  
    Hello  
</h1>
```

Как мы видим, все работает

Выводы

В данной работе мы с использованием wordpress и php сделали форму для записи товаров в бд и модифицировали стандартный механизм авторизации. Так же на питоне был написан веб сервер. Все цели были достигнуты