SILESIAN UNIVERSITY OF TECHNOLOGY FACULTY OF AUTOMATIC CONTROL, ELECTRONICS AND COMPUTER SCIENCE - FinTax - Raport 6.

Kacper Szymaniak Bartłomiej Murmyłowski
Group: 5TI Section: 533

7 Novemver 2023

Table Of Content

1 Introduction

This week, we focused on creating additional fundamental components for connecting our website to the database through PHP.

2 PHP

We implemented the PHP code to establish a connection to the "FinTax" database located on localhost. The decision to use PHP for this task is rooted in its widespread use as a server-side scripting language, particularly well-suited for web development. PHP seamlessly integrates with MySQL databases, making it a popular choice for dynamic web applications.

The code follows best practices for secure database connections, utilizing the MySQLi extension. It incorporates variables for storing sensitive information such as the database username and password, promoting a modular and secure approach.

Furthermore, the script includes error handling to gracefully manage connection failures, providing meaningful error messages for troubleshooting. The successful connection is confirmed with an echoed message.

By encapsulating the database connection logic in a reusable script, we promote maintainability and code organization. This approach aligns with the principles of clean and efficient coding, facilitating future enhancements and scalability in the development process.

```
<?php
// Dane do  poczenia  z  baz  danych
$host = "localhost"; // Adres hosta
$username = "nazwa_uzytkownika"; // Nazwa uytkownika
$password = "haslo"; // Haso
$database = "FinTax"; // Nazwa bazy danych

// Tworzymy  poczenie  z  baz  danych
$conn = new mysqli($host, $username, $password, $database);

// Sprawdzamy czy  poczenie  udao  si   nawiza
if ($conn->connect_error) {
    die("Nie  udao  si    poczy   z  baz  danych: " . $conn->connect_error);
}

echo " Poczenie  z  baz  danych udane.";

// Tutaj moesz  doda  dalsz  logik  aplikacji  korzystajc  z  poczenia  z  baz  danych

// Zamykamy  poczenie  po zakoczeniu pracy
$conn->close();
?>
```

Description of the code (*connection.php*) :

Database Connection Details: In this section, variables are defined to store the necessary information for establishing a connection to the database, such as the host address, database username, password, and database name.

Creating a Connection: The mysqli class is utilized to create a connection to the MySQL database using the specified connection details.

Checking Connection: It checks whether the connection was successful. If not, an error message is displayed, and the script terminates.

Echo Connection Status: If the connection is successful, a message indicating successful database connection is echoed.

Additional Logic: Developers can add further application logic using the established database connection.

Closing Connection: Finally, the database connection is closed when it's no longer needed.

Below is the PHP code that imports the database connection from connection.php and allows for subscribing and unsubscribing users to the "newsletter" table:

```
<?php
include 'connection.php'; // Include the database connection

// Handle subscribe (sub) and unsubscribe (unsub) actions
if (isset($_GET['action']) && ($_GET['action'] == 'sub' || $_GET['action'] == 'unsub')) {
    $email = $_GET['email']; // Assuming email is provided through the URL

    // Perform action based on subscribe or unsubscribe
    if ($_GET['action'] == 'sub') {
        subscribeUser($conn, $email);
    } else {
        unsubscribeUser($conn, $email);
    }
} else {
    echo "Invalid action specified.";
}

// Close the database connection
$conn->close();

// Subscribe user to the newsletter
function subscribeUser($conn, $email) {
    $dateSubbed = date('Y-m-d H:i:s'); // Get the current date and time

    // Insert subscriber into the database
    $sql = "INSERT INTO newsletter (email, date_subbed) VALUES ('$email', '$dateSubbed')";

    if ($conn->query($sql) === TRUE) {
        echo "User subscribed successfully.";
    } else {
        echo "Error subscribing user: " . $conn->error;
    }
}

// Unsubscribe user from the newsletter
function unsubscribeUser($conn, $email) {
    // Delete subscriber from the database
    $sql = "DELETE FROM newsletter WHERE email = '$email'";

    if ($conn->query($sql) === TRUE) {
        echo "User unsubscribed successfully.";
    } else {
        echo "Error unsubscribing user: " . $conn->error;
    }
}
?>
```

Description of the code (*newsletter.php*) :

- This file includes the connection.php file to utilize the established database connection.
- It handles subscription (sub) and unsubscription (unsub) actions based on the provided parameters in the URL.
- If the action is 'sub', it subscribes a user by inserting their email and the current date into the "newsletter" table.
- If the action is 'unsub', it unsubscribes a user by deleting their data from the "newsletter" table.
- Error messages are displayed in case of any issues during the subscription or unsubscription process.
- It's essential to have proper validation and sanitation of user inputs to prevent potential security vulnerabilities, especially in a production environment.

We are also working on PHP code that will be used for user registration and login.

3 Newsletter

We have created a newsletter table in the database using the following SQL code:

```
CREATE TABLE `newsletter` (
  `id` int(11) NOT NULL,
  `email` varchar(255) NOT NULL,
  `date_subbed` datetime NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

This code defines a table named newsletter with three columns:

1. **id:** An integer column that serves as the primary key and is set to auto-increment (int(11) NOT NULL).
2. **email:** A variable character column with a maximum length of 255 characters, representing the email address of the subscriber (varchar(255) NOT NULL).
3. **date subbed:** A datetime column to store the date and time when a user subscribes to the newsletter (datetime NOT NULL).

The table is set to use the InnoDB storage engine, which is a transaction-safe storage engine for MySQL, and it uses the UTF-8 character set.

This structure is suitable for a newsletter for several reasons:

Unique Identifier: The id column serves as a unique identifier for each record, ensuring that each entry is distinct.

Email Storage: The email column is designed to store email addresses, and its length is set to 255 characters, which is a common maximum length for email addresses.

Subscription Timestamp: The date subbed column captures the date and time of subscription. This is valuable for tracking when users subscribe, allowing for analytics and personalized communications.

InnoDB Engine: The use of the InnoDB storage engine provides transactional support, ensuring data integrity and reliability.

UTF-8 Character Set: The table is configured with the UTF-8 character set, enabling the storage of a wide range of characters, important for handling various languages and special characters in email addresses.

Overall, this table structure is well-suited for efficiently managing and retrieving newsletter subscription data while ensuring data accuracy and compatibility with different character sets.

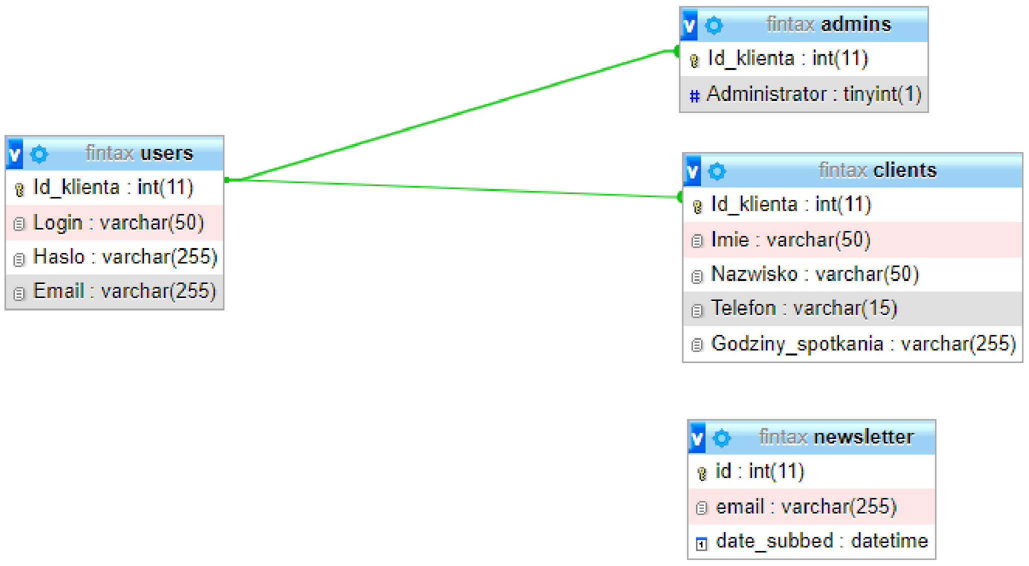

Figure 1: How does our current database look like.