МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЧЕРНІГІВСЬКА ПОЛІТЕХНІКА»
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ЕЛЕКТРОННИХ ТА
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Програмування Інтернет-систем

Методичні вказівки
до виконання лабораторних робіт та самостійної роботи
для здобувачів вищої освіти
спеціальності 121 – "**Інженерія програмного забезпечення**"
рівень вищої освіти – перший (бакалаврський)

Обговорено і рекомендовано
на засіданні кафедри інформаційних
технологій та програмної інженерії

Протокол №1
від 31 серпня 2021 р.

Чернігів, НУ «Чернігівська політехніка», 2021

Програмування Інтернет-систем. Методичні вказівки до виконання лабораторних робіт та самостійної роботи для здобувачів вищої освіти спеціальності 121 – «Інженерія програмного забезпечення», рівень вищої освіти – перший (бакалаврський) / Укладачі: Акименко А.М., Нехай В.В. – НУ «Чернігівська політехніка», 2021 р. – 67 с.

Укладачі: Акименко Андрій Миколайович, к.ф.-м.н., доцент кафедри ІТіПІ НУ «Чернігівська політехніка»;
Нехай Валентин Валентинович, старший викладач кафедри ІТіПІ НУ «Чернігівська політехніка».

Відповідальний за випуск: Трунова Олена Василівна, к.пед.н., доцент кафедри ІТіПІ НУ «Чернігівська політехніка»

Рецензент: Зайцев Сергій Васильович, професор кафедри інформаційних та комп'ютерних систем НУ «Чернігівська політехніка», д.т.н., професор

Зміст

Вступ

В результаті вивчення курсу, проведеного під керівництвом викладача, студенти оволодіють базовими теоретичним знаннями і практичними навичками, необхідними для розробки веб-додатків на мові програмування PHP, а також навичками роботи з системою управління базами даних MySQL. Також студенти познайомляться з основними принципами оптимізації сайтів, питаннями розміщення і управління контентом веб-додатків.

Мета курсу

Метою курсу є вивчення основних можливостей мови програмування PHP, принципів взаємодії з базами даних на прикладі MySQL, а також огляд основних принципів оптимізації сайту.

Після вивчення курсу слухачі зможуть:

- встановлювати і налаштовувати веб-сервер Apache, сервер даних MySQL, платформу PHP.
- розробляти базові веб-додатки;
- реалізовувати підключення веб-додатки до бази даних з метою зберігання та обміну інформацією між базою даних і додатком;
- працювати з веб-інтерфейсом MySQL PhpMyAdmin;
- використовувати методи GET і POST для передачі і обміну даними;
- використовувати HTML - Форми для забезпечення вводу, виводу і обробки даних веб-додатків;
- реалізовувати роботу з файлами і каталогами;
- використовувати основні принципи адміністрування веб-додатків.

У ході роботи необхідно розробити сайт під назвою «MyTravelNotes», що містить записи автора сайту про його подорожі, а також базу даних (БД) під назвою «MySiteDB», що містить контент сайту. Дана задача включає в себе реалізацію наступних функцій веб - проекту:

1. Можливість додавання записів автора і коментарів до них (При цьому всі замітки і коментарі повинні передаватися і зберігатися на сервері в БД);

2. Можливість модифікації і видалення нотаток і коментарів в БД допомогою форм сайту;

3. Забезпечення захисту даних за допомогою логіна і пароля, розмежування доступу до даних з урахуванням встановленого рівня доступу (Адміністратор і користувач);

4. Дані про користувачах і їх рівні доступу повинні зберігатися на сервері в БД;

5. Забезпечення коректного входу і виходу з сайту;

6. Реалізацію можливості зворотного зв'язку відвідувачів сайту з автором блогу;

7. Забезпечення дружнього користувальницького інтерфейсу і коректної організації навігації по розділам сайту.

Крім того, при розробці даного програмного проекту повинні враховуватися основні принципи web-дизайну (юзабіліті) для зручності роботи з сайтом кінцевого користувача.

Лабораторна робота №1: Установка набору дистрибутивів Denwer

У даній лабораторній роботі розглядаються установка набору дистрибутивів, необхідних для розробки серверних додатків за допомогою мови програмування PHP і настройка інструментального середовища Adobe Dreamweaver на роботу з віртуальним сервером.

Вправа 1: Установка набору дистрибутивів Denwer

У даній вправі буде продемонстрована установка набору дистрибутивів, до складу якого входять дистрибутиви Apache, PHP, MySQL, phpMyAdmin і інших систем і додатків, необхідних для організації розробки серверних додатків з використанням засобів мови програмування PHP.

1. Завантажте дистрибутив базового комплекту Denwer - Denwer_base (близько 2Мб), найостаннішу версію комплекту Denwer можна взяти з сайту - www.denwer.ru/dis/Base.

2. Запустіть інсталятор. Спочатку архів буде автоматично розпакований в тимчасову директорію, а потім автоматично запуститься інсталятор.

3. За замовчуванням для установки комплексу використовується директорія C: \ WebServers, натисніть Enter, щоб погодитися з цим вибором. У зазначеному каталозі будуть розташовані абсолютно всі компоненти системи, і поза ним ніякі файли в подальшому не створюються (виключаючи ярлики на Робочому столі).

4. Далі вам запропонують ввести ім'я віртуального диску, який буде пов'язаний з тільки що зазначеною Директорією. Рекомендується погодитися зі значенням за замовчуванням (Z :). Важливо, що диску з цим ім'ям ще не повинно міститися в системі - найчастіше так і відбувається з диском Z :.

5. Після цього почнеться копіювання файлів дистрибутиву. У ході завантаження вам буде запропонована виведення ярликів на Робочий стіл - це необхідно зробити для подальшого зручності роботи.

6. Після закінчення завантаження з'явиться запит про те, як саме ви збираєтеся запускати і зупиняти комплекс (2 варіанти):

   - Автоматично створювати віртуальний диск при завантаженні машини (а при зупинці серверів цей диск не відключати);
   - Створювати віртуальний диск тільки по явною команді старту комплексу (при натисканні на ярлику запуску на робочому столі). І, відповідно, відключати диск від системи - при зупинці серверів. **Рекомендований варіант - другий.**

7. Завантаження завершено. На Робочому столі Windows двічі клацніть на

ярличок Start Denwer (якщо ви не створювали ярлики, то можна запустити Денвер по команді C: \ WebServers / denwer \ Run.exe).

8. Дочекавшись, коли всі консольні вікна зникнуть, відкрийте браузер і наберіть в ньому адресу: http: // localhost. У разі успішного встановлення відкриється сторінка, яка повідомляє про це (див. Рис. 1.1).

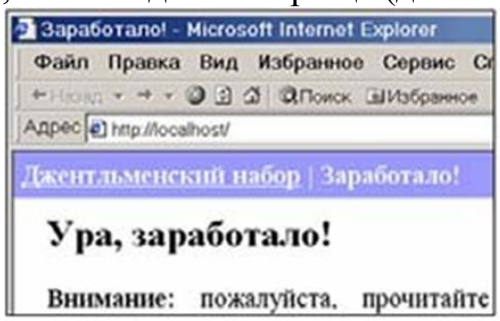

Рис.1.1. Вікно сторінки http://localhost

9. Вивчіть інформацію сторінки http://localhost/ Зверніть особливу увагу на розділи присвячені тестуванню і роботі з наявними утилітами.

Вправа 2: Налаштування Adobe Dreamweaver на роботу з сервером Apache

У даній вправі буде продемонстрована налаштування інструментального середовища розробки Adobe Dreamweaver для роботи з віртуальним сервером Apache.

1. Запустіть програму *Adobe Dreamweaver*.
2. Виберіть пункти меню **Веб-сайт - Управління веб-сайтами**. Відкриється діалогове вікно «Керування сайтами». В ньому натисніть кнопку «Створити». Відкриється вікно налаштування сайту. На вкладці **Веб-сайт** вікна налаштування сайту введіть:
   - ім'я сайту: «*MyTravelNotes»;*
     вкажіть шлях до локальної папки його розташування:
     C:\WebServers\home\localhost\www
3. У тому ж вікні перейдіть на вкладку **Сервери**, де необхідно задати параметри віртуального сервера. Для цього натисніть «+». Відкриється вікно налаштувань сервера.
4. На вкладці **Базовий** вкажіть:
   - Ім'я серверу: *localhost*
   - Підключення за допомогою: *Локальний / Мережевий*
   - Папка серверу: *C:\WebServers\home\localhost\www*
   - URL - адреса: *http://localhost*
5. У тому ж вікні перейдіть до вкладки **Додатково**:
   - У розділі **Віддалений сервер** встановіть галочку напроти пункту *«Зберегти відомості про синхронізацію».*
   - У розділі **Тестовий сервер** з меню, що випадає **Серверна модель** виберіть *PHP MySQL*.

6. Збережіть параметри. Зверніть увагу, що у вікні «Налаштування сайту» на вкладці Сервери повинні стояти два прапори для localhost: **Віддалений** і **Тестовий** (якщо їх немає - натисніть їх самостійно).
7. У вікні **Управління сайтами** з'явиться ім'я створеного проекту сайту «***MyTravelNotes***». Натисніть кнопку **Готово**. Для корегування, видалення та інших операцій з проектом сайту використовуйте відповідні кнопки управління в даному вікні (*рис. 1.2*).

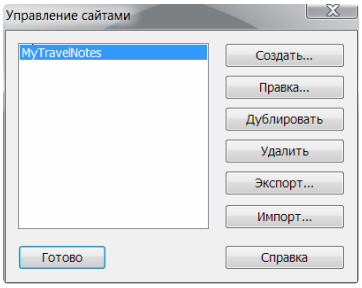

Рис. 1.2. Вікно управління проектом сайту.

Вправа 3: Перша програма на PHP

У цій вправі Ви напишіть програму "Hello, world!" мовою PHP, використовуючи середовище розробки Adobe Dreamweaver.

1. Створіть у програмі Adobe Dreamweaver нову сторінку (**Файл–Створити.** Виберіть тип сторінки PHP).
2. Введіть наступний PHP-код:

```
<BODY>
<?PHP
echo "Hello, world!";
?>
</BODY>
```

3. Збережіть сценарій в папку сайту під ім'ям hello.php.
4. Перевірте результат в браузері (F12).

При перегляді результату виконання файлу hello.php в браузері перегляньте html-код (*наприклад, в IE меню* ***Вид - Перегляд HTML-коду***). Зверніть увагу на те, що php-коду на сторінці немає - це означає, що php-сценарій був оброблений сервером, після чого сервер передав в браузер результат обробленого php-сценарію.

Вправа 4: Найпростіші програми на PHP

Далі представлені додаткові вправи для закріплення навичок створення найпростіших програм в середовищі Adobe Dreamweaver на мові програмування PHP. Для їх виконання в Adobe Dreamweaver створіть новий .php файл (**Файл - Створити - Вибрати тип файлу** .php) **examples.php** і виконайте запропоновані далі завдання.

1. Змінній $a необхідно присвоїти значення 10, змінній $b присвоїти значення 20. Виведіть значення змінних на екран.
2. Потім змінній $c надайте значення суми цих змінних (змінної $a і змінної $b). Виведіть значення змінної $c на екран.
3. Далі збільште значення змінної $c в три рази і виведіть отриманий результат на екран.
4. Розділіть змінну $c на різницю змінних $b і $a, виведіть результат на екран.
5. Введіть нові змінні $p і $b. Дайте змінної $p значення «Програма», а змінній $b значення «працює».
6. Потім складіть змінні, що містять ці слова («Програма» і «працює»), при цьому слова повинні бути розділені пропуском (' '). Результат необхідно привласнити змінній $result.
7. Далі за допомогою оператора «.=» Необхідно до рядка «Програма працює» додасть слово «добре». Результат необхідно привласнити змінній $result.
8. Є дві змінні: $q = 5 і $w = 7. Створіть скрипт, в результаті виконання якого ці дві змінні «обмінюються» значеннями - змінна $q отримує значення 7, змінна $w отримує значення 5, при цьому не створюючи нових змінних (варіант $q = 7 і $w = 5 не розглядається).

Лабораторна робота №2: Створення статичного каркасу сайту. Робота з інструментарієм середовища розробки Adobe Dreamweaver

У даній лабораторній роботі ілюструється створення двох перших сторінок сайту – **blog.html** та **inform.html** в статичному вигляді за допомогою засобів розробки інструментального середовища Adobe Dreamweaver. Сторінка **blog.html** є першою сторінкою сайту, повинна завантажуватися в браузері і містити власне замітки автора блогу. Сторінка статистики та **inform.html** буде допоміжною сторінкою, що містить статистичну інформацію про розміщені на сайті нотатках і коментарях.

Вправа 1: Налаштування Adobe Dreamweaver

У даній вправі продемонстрований процес створення web-проекту в інструментальному середовищі Adobe Dreamweaver.

1. Запустіть Denwer.
2. Запуск Adobe Dreamweaver, відкрийте багатофункціональний вікно **Налаштування (Правка - Налаштування)**, категорія **Створити документ**. Встановіть тип документа за замовчуванням (DTD) - HTML5 і кодування за замовчуванням - кирилиця (Windows).

Вправа 2: Створення статичної основи web-сторінок

У даній вправі необхідно створити дві перші статичні сторінки проекту, які стануть основою для подальшої розробки. Сторінка **blog.html** є першою сторінкою сайту, повинна завантажуватися в браузері і містити власне замітки автора блогу. Сторінка статистики **inform.html** буде допоміжною сторінкою, що містить статистичну інформацію про розміщені на сайті нотатках і коментарях.

1. В Dreamweaver перейдіть в меню **Файл - Створити**.
2. Створіть статичний документ в форматі HTML для майбутньої сторінки з назвою **blog.html**. На сторінці повинно розташовуватися меню переходів між сторінками і місце розміщення основного контенту сайту (рис.2.1):

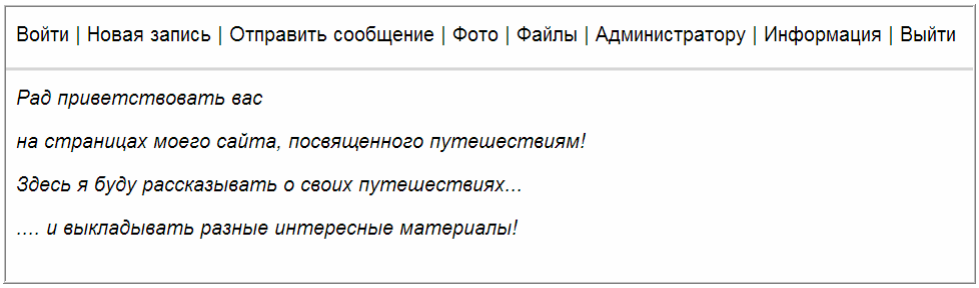

Рис. 2.1. Приклад сторінки blog.html

3. Збережіть сторінку під ім'ям blog.html.
4. творіть сторінку статистики inform.html. Схема сторінки (рис. 2.2):

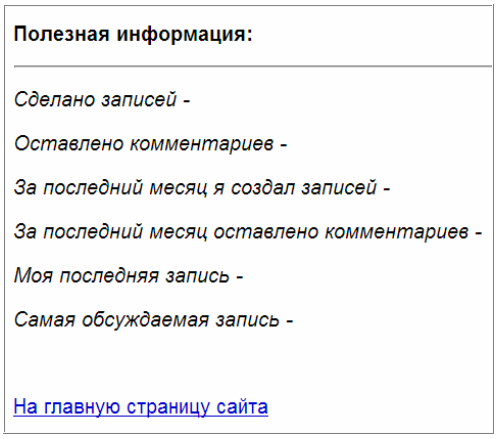

Рис. 2.2. Приклад сторінки inform.html

5. Збережіть таблицю під ім'ям inform.html.
6. Зв’яжіть гіперпосиланнями створені сторінки (меню **Змінити - Створити посилання** або вручну засобами мови розмітки HTML).

Лабораторна робота №3: Створення бази даних MySQL

У ході виконання даної лабораторної роботи необхідно створити в MySQL нову базу даних з назвою «***MySiteDB***» і додати до неї дві таблиці: **notes** та **comments**. **Notes** містить замітки блогу; **comments** – коментарі до цих нотаток. Схема даних (*рис.3.1*):

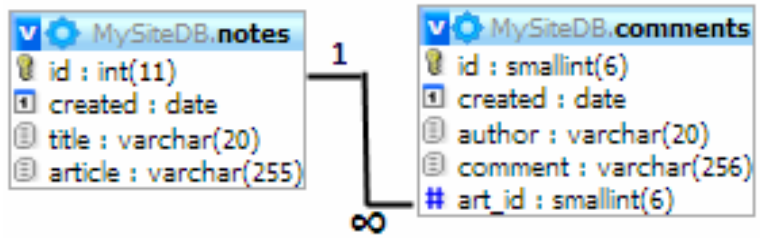

Рис.3.1. Схема бази даних "MySiteDB"

У *Додатку 4* представлена інформація про основні поняття, необхідні для роботи з базою даних.

Вправа 1: Створення БД «MySiteDB»

У цій вправі реалізується запит на створення нової бази даних.

1. Створіть новий php документ, який буде називатися **create_db.php.**
2. Створіть з'єднання з сервером localhost. Ім'я сервера ***localhost***, користувач ***root***, пароля немає.
3. Створіть базу даних:
   3.1. Сформуйте запит на створення бази ***MySiteDB*** з використанням SQL;
   3.2. Реалізуйте запит на створення БД за допомогою функції **mysqli_query ().**
4. Зберегти документ, виконати запит.
5. За допомогою утиліти **PhpMyAdmin** переконайтесь, що створена нова база даних. Для цього запустіть утиліту: http://localhost/tools/phpmyadmin (або http://localhost і виберіть PhpMyAdmin зі списку утиліт).
6. Вдруге виконайте запит, щоб переконатися, що з'єднання є, а база не створюється (тому що вона була вже створена раніше, в ході попереднього виконання скрипту).
7. Бажано додати цикл if для виявлення неполадок в роботі. Варіант реалізації створення БД MySiteDB

```
<?php
// Створити з'єднання з сервером
$link = mysqli_connect ("localhost", "root", "");
if ($link) {
   echo " З'єднання з сервером встановлено ", "<br>";
   } else {
         echo " Немає з'єднання з сервером ";
                  }
```

```
// Створити БД MySiteDB
// Спочатку формування запиту на створення
$db = "MySiteDB";
$query = "CREATE DATABASE $db";

// Потім реалізація запиту на створення. Важлива послідовність
аргументів функції: з'єднання з сервером, SQL-запит.
$create_db = mysqli_query($link, $query);
if ($create_db) {
   echo " База даних $db успішно створена ";
   } else {
         echo " База не створена ";
               }
?>
```

Вправа 2: Створення користувача admin

У цій вправі необхідно створити нового користувача бази даних з ім'ям **admin** і паролем **admin** з правами адміністратора. Користувачів можна додавати двома способами:

- за допомогою SQL-запиту GRANT
- в таблиці призначення привілеїв MySQL (Privileges) за допомогою утиліти PhpMyAdmin.

Виберіть один з двох наведених далі способів.

Спосіб 1: створення нового користувача за допомогою SQL-запиту GRANT

1. Створіть новий php-документ, який буде називатися **create_user.php;**
2. Створіть з'єднання з сервером;
3. Сформуйте SQL-запит на створення нового користувача бази даних:

```
$query = "GRANT ALL PRIVILEGES ON *.* TO 'admin'@'localhost'
IDENTIFIED BY 'admin'
            WITH GRANT OPTION";
```

//*.* - *глобальний рівень привілеїв, застосовується до всіх баз на сервері.*

4. Реалізуйте запит.

- Перевірка створення користувача. За допомогою утиліти PhpMyAdmin переконайтесь, що створений новий користувач. Для цього запустіть утиліту PhpMyAdmin і перейдіть до вкладки Privileges. Вивчіть список користувачів.

Спосіб 2: створення нового користувача за допомогою утиліти PhpMyAdmin

1. Запустіть утиліту PhpMyAdmin і перейдіть до вкладки Privileges.

Натисніть кнопку «Add a new user».

2. Введіть ім'я користувача (***admin***), ім'я сервера (***localhost***), пароль з підтвердженням (***admin***). Надайте новому користувачу всі права (***global privileges – Check All***).
3. Переконайтесь, що новий користувач створений коректно.
4. Всі подальші дії з базою даних будуть проводитися під користувачем **admin** з паролем **admin** і відповідними правами, якщо інше не зазначено в завданні.

Вправа 3: Створення таблиці notes

У даній вправі буде продемонстрований один із способів створення таблиць в раніше створеній базі даних на прикладі створення таблиці ***notes***. Таблиця ***notes*** містить замітки автора блогу. Дана таблиця буде створена засобами PHP. Інформацію про поля таблиці см. В *Додатку 3*.

1. Створіть новий php-документ, який буде називатися create_tbl.php**;**
2. Створіть з'єднання з сервером вже під створеним раніше користувачем ***admin*** з паролем ***admin***.
3. Підключіться до бази даних MySiteDB.
4. Сформуйте запит на створення таблиці **notes** з полями, зазначеними в *Додатку 3*.

```
// Формування запиту
$query = "CREATE TABLE notes
                                (id INT NOT NULL AUTO_INCREMENT,
                                PRIMARY KEY (id),
                                 created DATE,
                                 title VARCHAR (20),
                                article VARCHAR (255))";
```

5. Реалізуйте запит на створення таблиці.
6. За допомогою утиліти PhpMyAdmin переконайтесь, що створено нову таблицю. Для цього запустіть утиліту, перейдіть до бази даних MySiteDB і перегляньте її структуру. У ній повинна з'явитися відповідна таблиця.

Варіант реалізації створення таблиці notes

```
<?php
// З'єднання з сервером
$link = mysqli_connect ('localhost', 'admin', 'admin');

//Вибір БД
$db = "mySiteDB";
$select = mysqli_select_db($link, $db);
if ($select){
```

```
        echo " База успішно обрана ", "<br>";
        } else {
                echo " База не обрана ";
        }
// Створення таблиці
// Формування запиту
$query = "CREATE TABLE notes
                                (id INT NOT NULL AUTO_INCREMENT,
                                PRIMARY KEY (id),
                                 created DATE,
                                 title VARCHAR (20),
                                article VARCHAR (255))";
// Реалізація запиту
$create_tbl = mysqli_query ($link, $query);
if ($create_tbl) {
        echo " Таблиця успішно створена", "<br>";
        } else {
                echo " Таблиця не створена";
        }
?>
```

Вправа 4: Створення таблиці comments

У даній вправі буде продемонстрований інший спосіб створення таблиць в раніше створеній базі даних на прикладі створення таблиці **comments**. Таблиця **comments** містить коментарі користувачів до нотаток автора блогу. Таблиця буде створена за допомогою утиліти PhpMyAdmin. Інформацію про поля таблиці см. В *Додатку 3*.

1. Запустіть браузер.
2. Запустіть утиліту phpMyAdmin. У головному вікні PHPMyAdmin виберіть БД MySiteDB.
3. В поле "Create new table", назвіть таблиці - comments; кількість полів - 5, натисніть кнопку «Go» (*рис. 3.2*).

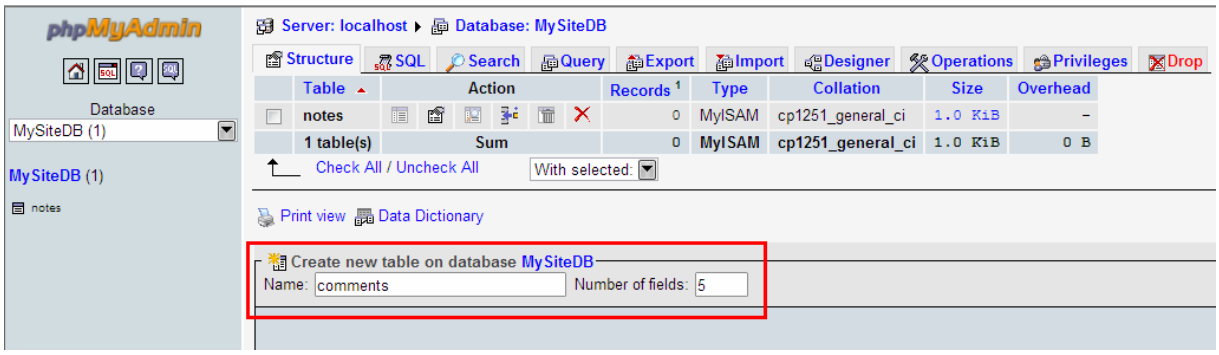

Рис. 3.2. Створення нової таблиці за допомогою утиліти phpMyAdmin.

4. Створення полів таблиці comments:
   4.1. У вікні, заповніть необхідні поля таблиці (рис. 3.3) і натисніть кнопку

«Save».

4.2. Для поля id додайте наступні атрибути: позначте автоінкремент A_I і первинний ключ PRIMARY в поле зі списком INDEX.

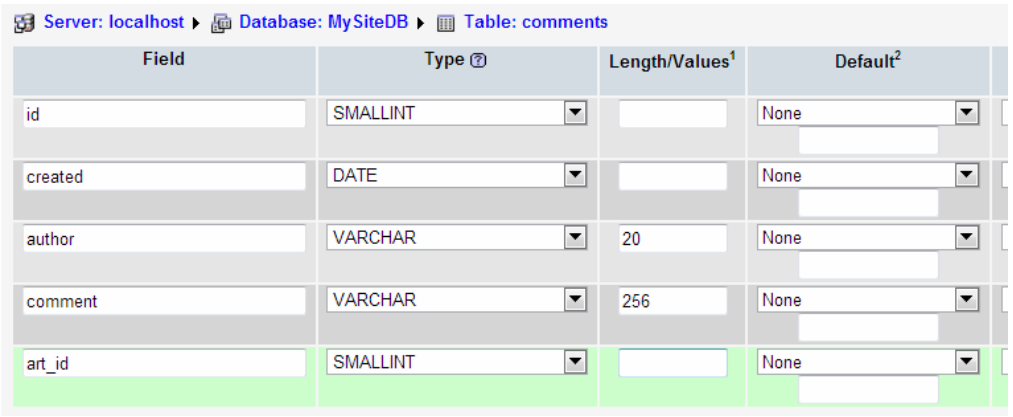

Рис. 3.3. Заповнення полів таблиці

5. Отриманий результат повинен виглядати наступним чином (*рис. 3.4*):

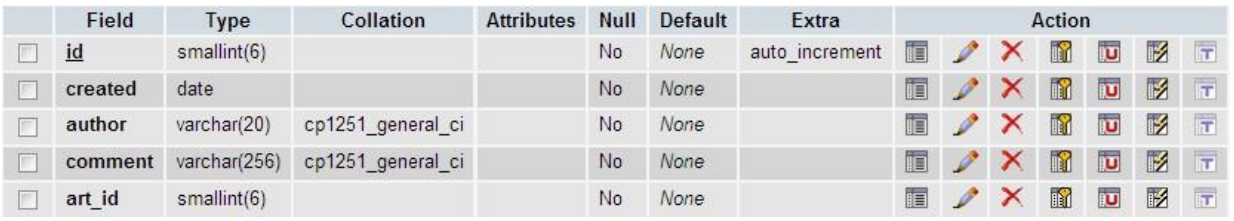

Рис. 3.4. Результат створення таблиці

Вправа 5: Створення між табличних зв'язків

У даній вправі необхідно створити зв'язки між таблицями для підтримки цілісності даних web-додатки.

1. Для організації між табличних зв'язків виберіть БД MySiteDB, вкладку Designer. Відкриється вікно схеми даних.
2. За допомогою інструментів вікна Designer створіть зв'язок «один до багатьох» *(рис. 3.5*).

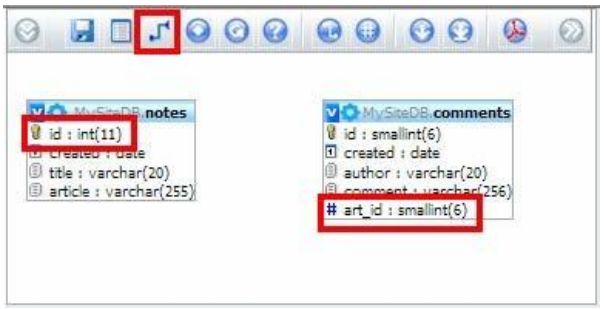

Рис. 3.5 Поле вікна інструментів Designer

3. Введіть в створені таблиці кілька записів - для перевірки їх роботи і для використання на майбутніх серверних сторінках сайту. Для цього виберіть потрібну таблицю і натисніть кнопку **Insert** у групі **Action**. Після заповнення відповідних полів таблиці натисніть кнопку **Go**.

- *Пам'ятайте, що поля **id** в таблицях заповнювати не треба - вони заповнюються автоматично.*

- *Поле **art_id** таблиці **comments** має бути прив'язане до поля **id** таблиці **notes**, тому коментарі створюються тільки в прив'язці до конкретної замітки. MySQL автоматично в поле **art_id** таблиці **comments** підставляє список що випадає **id** вже створених нотатках, вам необхідно лише вибрати **id** замітки зі списку.*

Вправа 6: Реєстрація бази даних в Adobe Dreamweaver для підключення до сайту

Реєстрація включає створення імені та пароля користувача, від імені якого ведеться робота з базою (в нашому випадку **admin**), а також адреса сервера даних (**localhost**) і ім'я бази (**MySiteDB**).

1. Запустіть сервер і базу даних MySQL.
2. У Adobe Dreamweaver в меню **Бази даних** натисніть кнопку «плюс (+)». З'явиться єдиний пункт - **Підключення MySQL**.
3. Далі з'явиться діалогове вікно, яке необхідно заповнити (див. *Рис.3.6*). Останній розділ «База даних» можна заповнити як самостійно, так і натиснувши **Обрати**. Протестуйте з'єднання. потім натисніть **ОК**.

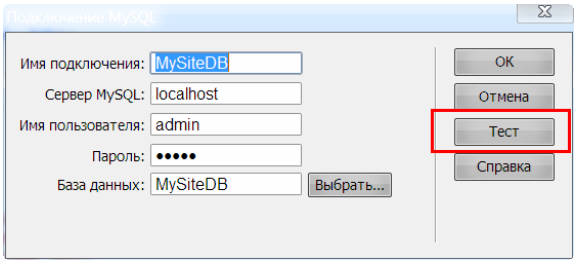

Рис. 3.6. Заповнення полів вікна підключення бази даних до проекту

4. Після підключення в меню Бази даних повинна відобразитися підключена нами БД MySiteDB з двома таблицями (*рис. 3.7*).

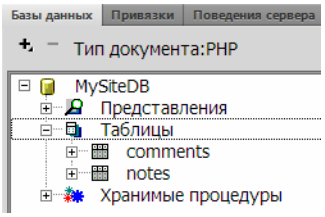

Рис. 3.7. Відображення підключеної до проекту бази даних

5. Переконайтесь, що в меню **Файли** з'явилася нова папка **Connections** і в ній файл **MySiteDB.php** (Назва файлу збігається з ім'ям з'єднання).

Вправа 7: Файл підключення бази даних

В ході виконання даної вправи необхідно внести зміни в код

автоматично створеного файлу підключення бази даних для настройки коректної роботи підключеної бази даних з кирилицею.

1. Відкрийте файл MySiteDB.php і внесіть в нього такі зміни:
   1.1. Змініть найменування змінних для зручності подальшої роботи;
   1.2. Змініть функцію ***mysql_pconnect()*** на ***mysqli_connect();***
   1.3. Внесіть доповнення для коректної кодування символів в базі даних:

```
<?php
# FileName="Connection_php_mysql.htm"
# Type="MYSQL"
# HTTP="true"
$localhost = "localhost";
$db = "MySiteDB";
$user = "admin";
$password = "admin";

$link = mysqli_connect($localhost, $user, $password) or
trigger_error(mysql_error(),E_USER_ERROR);
//trigger_error виводить на сторінку повідомлення про помилку. перший
параметр
- повідомлення про помилку
//в строковому вигляді, в даному випадку повертається функціонально
mysql_error(), другий - числовий код // помилки (майже завжди
використовується значення константи E_USER_ERROR, равное 256)

// Наступні рядки необхідні для того, щоб MySQL сприймав кирилицю.
// Параметри функції mysqli_query (): ідентифікатор з'єднання з сервером і
запит SQL

mysqli_query($link, "SET NAMES cp1251;") or die(mysql_error());
mysqli_query($link, "SET CHARACTER SET cp1251;") or die(mysql_error());
?>
```

Лабораторна робота №4: Просте виведення даних. Сторінки blog.php і comments.php.

Вправа 1: Вивід даних з бази на сторінку

У цій вправі на головну сторінку сайту необхідно вивести всі нотатки з таблиці БД **notes**.

1. Перейменуйте **blog.html** в **blog.php** (стару html-сторінку після перейменування можна видалити).
2. Видаліть записи на сторінці. Повинні залишитися тільки привітання і навігаційне меню.
3. Створіть з'єднання з сервером. Воно у нас вже реалізовано в файлі **MySitDB.php** – файл треба просто включити за допомогою функції ***require_once(),*** в якості параметра передавши їй шлях до файлу («**Connections/MySiteBD.php**»):

   <?php require_once ("connections/MySiteDB.php"); ?>

4. Далі необхідно вивести записи (рядки) на сторінку сайту з таблиці **notes**. Спочатку треба реалізувати запит на вибірку. Для цього:

   4.1. Оберіть БД;
   4.2. Створіть SQL-запит на вибірку даних з таблиці (SELECT fields FROM tableName). Тут SELECT - оператор вибору полів, FROM - оператор вибору таблиці-джерела полів.

   - *Якщо вам необхідно вибрати все поля таблиці (як в даному випадку), то запит можна побудувати так: SELECT * FROM tablename, де символ «*» позначає все поля таблиці.*

   4.3. Реалізуйте запит на вибірку.

5. Далі необхідно вивести запис на сторінку сайту. Для цього використовується функція ***mysqli_fetch_array()***. Параметром функції є змінна, що містить результат виконання запиту до БД (в даному випадку - реалізації запиту на вибірку); власне функція отримує по одному запису з таблиці за один раз. Кожен запис повертається у вигляді масиву.

6. Для виведення інформації з масиву по окремих елементах необхідно дотримуватися наступного синтаксису:

```
// Висновок елементів масиву
echo $note ['id'], "<br>";
echo $note ['created'], "<br>";
```

```
echo $note ['title'], "<br>"; echo
$note ['article'], "<br>";
```

7. Зараз з таблиці за допомогою функції ***mysqli_fetch_array()*** виводиться тільки один запис. За допомогою циклу необхідно зробити так, щоб виводилися всі записи з таблиці. Для цього необхідно змінити частину коду таким чином:

```
// Використання циклу while
while ($note = mysqli_fetch_array($select_note)){
        echo $note ['id'], "<br>";
        echo $note ['created'], "<br>";
        echo $note ['title'], "<br>";
        echo $note ['article'], "<br>";}
```

Тут змінної з ім'ям *$select_note* присвоюється результат виконання запиту до БД *mysqli_query ().*

Вправа 2: Обмін даними між серверними сторінками

Кожна замітка на головній сторінці блогу може бути прокоментована. Для реалізації цієї функції необхідно зробити з заголовка кожної замітки гіперпосилання, перейшовши по якій відвідувач потрапить на сторінку зі списком коментарів до обраної замітці. Крім того, на цій же сторінці повинна відображатися сама обрана для коментування замітка.

Отже, необхідно реалізувати механізм обміну даними між сторінками таким чином, щоб при переході по посиланню передавалася інформація про те, яка саме замітка була обрана.

Для цього необхідно ввести якийсь ідентифікатор, значення якого буде збігатися з id коментованій замітки, і який буде передаватися при переході по посиланню.

1. Створення гіперпосилання

1.1. Створіть нову сторінку comments.php, яка буде містити коментарі до обраної замітки.
1.2. Реалізуйте з'єднання з сервером.
1.3. Реалізуйте підключення до БД.
1.4. Для передачі ідентифікатора замітки введемо аргумент **note**. Як значення він буде отримувати значення поля ***id*** таблиці **notes**.
1.5. На сторінці **blog.php** знайдіть фрагмент коду, що передає заголовок замітки ***title (echo $note [‘title’];)***. Його необхідно відредагувати таким чином, щоб він став гіперпосиланням на сторінку коментарів **comments.php**, а також передавав ***id*** вибрані нотатки:

```
while ($note = mysqli_fetch_array($select_note)){ echo $note['id'],
"<br>";
?>
```

```
<a href="comments.php?note=<?php echo $note['id']; ?>">
<?php echo $note ['title'], "<br>";?></a>

<?php

echo $note ['created'], "<br>"; echo $note ['content'], "<br>";

}
?>
```

Тут ми створюємо гіперпосилання на сторінку **comments.php** і в цим гіперпосиланням передаємо ідентифікатор **note**, значення якого дорівнює значенню елемента масиву ***$note['id'],*** тобто значенням ***id*** нотатки.

2. Сторінка **comments.php**

2.1. Перейдіть на сторінку **comments.php**. На даній сторінці повинні відображатися коментарі до обраного запису, а також сам коментований запис (для зручності користувача сайту).

2.2. Дану задачу можна виконати за аналогією з висновком заміток на сторінці **blog.php**. Основна відмінність полягає в тому, що спочатку необхідно зі сторінки **blog.php** отримати переданий за допомогою ідентифікатора **note** id нотатки. Це робиться за допомогою методу$_GET:

```
// Змінної $note_id необхідно привласнити id замітки, переданої за
допомогою методу $_GET зі сторінки blog.php
$note_id = $_GET['note'];
```

2.3. Далі необхідно вивести значення полів created, title, content з таблиці **notes** для замітки з отриманим id. Для цього використовується SQL запит

SELECT… FROM… WHERE…

У ньому за допомогою оператора ***SELECT*** вибираємо необхідні поля таблиці; за допомогою ***FROM*** визначаємо таблицю-джерело вибірки; ***WHERE*** задає умову відбору, за яким вибираємо замітку з обраним ***id***:

```
// Формуємо SQL-запит на вибірку з урахуванням переданого id замітки
$query = "SELECT created, title, article FROM notes WHERE id = $note_id";
```

2.4. Після формування SQL-запиту його необхідно реалізувати за допомогою функції ***mysqli_query()*** і вивести дані на сторінку за допомогою функції ***mysqli_fetch_array().***

2.5. Потім аналогічним чином виведіть коментарі до обраної замітки. Зверніть увагу, що SQL-запит на вибірку коментарів повинен будуватися таким чином:

$query_comments = "SELECT * FROM comments WHERE art_id =$note_id";

В умові WHERE ми реалізуємо підтримку зв'язку між таблицями, які пов'язані за полями ***id*** (таблиця **notes**) та ***art_id*** (таблиця **comments**).

В змінній **$note_id** міститься ***id*** вибрані нотатки отже, для вибору коментарів до цієї статті необхідно, щоб значення поля ***art_id***created також дорівнювало **$note_id.**

3. Перевірте коректність даних між сторінками **blog.php** и **comments.php**. При переході за посиланням з **blog.php** на **comments.php** в адресному рядку браузера повинен відображатися ***id*** вибраного нотатку, переданий за допомогою ідентифікатора **note**.

4. Для того, щоб виводилися всі коментарі, а не тільки перший - реалізуйте цикл.

5. Якщо у замітки немає жодного коментаря - про це треба повідомити.
   5.1. Під областю коментарів додайте напис «*Цей запис ще ніхто не коментував*».
   5.2. У коді програми створіть цикли з умовою **if**: якщо хоча б один коментар існує - він повинен бути виведений (тобто елементи масиву повинні бути відображені); якщо кількість коментарів дорівнює нулю - повинна виводитися напис «*Цей запис ще ніхто не коментував*».

Лабораторна робота №5: Введення і правка даних за допомогою форми

У ході виконання даної лабораторної роботи розглядаються принципи введення інформації та модифікації контенту сайту за допомогою обробки форм.

Вправа 1: Відправка пошти

Ця вправа дозволяє реалізувати відправку повідомлення через форму на сервер.

1. Створіть сторінку **email.php**. Додайте назву сторінки і пояснювальний текст, форму з двома текстовими полями: **Тема повідомлення** і **Текст повідомлення**, кнопку **Відправити**, а також гіперпосилання для повернення на головну сторінку сайту.

2. Самостійно реалізуйте обробку даних форми за допомогою функції **mail().** «Отримати» відправлене повідомлення ви можете по локальній адресі: C:\WebServers\tmp\!sendmail\.

3. Перевірте коректність роботи, створіть гіперпосилання з головної сторінки сайту на сторінку email.php і зі сторінки email.php на сторінку blog.php.

4. Самостійно реалізуйте перевірку заповнення всіх полів форми для того, щоб виключити відправку «порожнього» листа.

Вправа 2: Сторінка для додавання нотаток

У цій вправі буде проілюстровано створення сторінки для додавання нових нотаток – **newnote.php**.

1. Створіть нову сторінку **newnote.php**, додайте назву і пояснювальний текст.

2. Створити html-форму з назвою «**newnote**», метод обробки даних – **POST.**

3. На формі розташуйте два поля: одне (типу *text*) для додавання заголовку нотатки – з назвою «**title**», друге (*textarea*) для додаваня самої нотатки – з назвою «**article**». Додайте параметри розміру елементів форми.

4. Також помістіть на поле кнопку відправки з назвою «**submit**».

! Не забувайте іменувати html-форму та елементи html-формы (атрибут **name**). Дані імена важливі для наступної обробки даних, отриманих через форму, в php-скриптах.

5. Додавання дати створення нотатки

5.1. У таблиці **notes**, що заповнюється через створювану нами форму, залишилося незаповненим поле **art_id** (поле з датою створення

нотатки) – для нього ми не створювали елемент форми. PHP дозволяє отримувати поточну дату автоматично, за допомогою функції ***date().*** Формат її виклику: ***date(<формат>).*** MySQL вимагає формат дати <рік>-<місяць>-<число>, при цьому рік – 4 цифри, місяць – 2 цифри, число – 2 цифри. Згідно з шаблоном, вид виклику функції: ***date("Y-m-d").*** Ми автоматизуємо процес отримання поточної дати з форми.

5.2. Розташуйте на формі після другого текстового поля приховане поле з назвою «***created***».

5.3. Значення поля ***created*** буде отримувати через php-функцію **date().**

Результат додавання поля:

<input type="hidden" name = "created" id = "created" value ="<?php echo date("Y-m-d");?>"/>

Варіант реалізації html-форми

```
<html>
<body>

<p>Додати нову нотатку: </p>
<form id="newnote" name="newnote" method="post">
<input type="text" name="title" id="title" size="20" maxlength="20"/>
<textarea name="article" cols="55" rows="10" id=" article"> </textarea>
<input type="hidden" name = "created" id = "created"
                value ="<?php echo date("Y-m-d");?>"/>
<input type="submit" name="submit" id="submit" value="Відправити" />
</form>
<a href="blog.php">Повернення на головну сторінку сайту</a>
</body>
</html>
```

6. Обробка html-форми. Вам необхідно створити php-скрипт, який виконає два кроки:

- Отримає дані, введені користувачем в поля створеної html-форми (тобто нову замітку)
- Відправить ці дані в базу, де зберігаються вже створені раніше нотатки.

6.1. Отримання даних через форму. Для отримання даних через форму необхідно:

6.1.1. Підключитися до сервера;
6.1.2. Обрати базу даних;
6.1.3. Отримати дані з полів форми. Дані ми отримуємо з елементів форми використовуючи назви (атрибут **name**) цих елементів. Дані форми поміщаються в масив $_POST, а потім присвоюються

змінним php. Принцип отримання:

$назва_змінної = $_POST *[‘АтрибутNameЕлементаФорми’*];

Таким чином інформація, введена користувачем у форму, «присвоюється» в якості значення для змінної php.

//Отримання даних з форми

$title = $_POST['title'];

$created = $_POST['created'];

$article = $_POST[‘article’];

6.2. Передача даних в базу

6.2.1. Дані в базу передаються по звичайному принципу: формування SQL-запиту - реалізація SQL-запиту. Формування запиту: (в ньому поле id отримує своє значення автоматично):

//Формування запиту

$query = "INSERT INTO notes (title, created, article)
VALUES ('$title', '$created', '$article)";

У запиті використовується SQL-інструкція INSERT. Синтаксис інструкції:

INSERT INTO tblName (tblField 1, tblField 2, ... , tblField N)
VALUES (value 1, value 2, ... , value N);

У ньому *tblNamt* - назва таблиці, *tblField* - назва поля таблиці (перераховуються в тому порядку, в якому розташовуються в таблиці), value – значення поля таблиці, яке вставляється (порядок повинен відповідати порядку назв полів).

6.2.2. Реалізуйте запит за допомогою функції mysqli_query().

7. Перевірте коректність роботи форми і обробки даних.
8. Самостійно програмно виключіть можливість передачі в базу даних порожнього запису.
9. Додайте гіперпосилання між сторінками **blog.php** та **newnote.php**.

Варіант реалізації коду сторінки newnote.php

```
<html>
<head>
    <title>Сторінка для додавання нотатки </title>
</head>

<body>
```

```
        <p>Додати нову нотатку: </p>
        <form id="newnote" name="newnote" method="post" action="">
        <input type="text" name="title" id="title" size="20" maxlength="20"/>
        <textarea name=" article" cols="55" rows="10" id=" article"> </textarea>
<input type="hidden" name = "created" id = "created" value
        ="<?php echo date("Y-m-d");?>"/>

    <input type="submit" name="submit" id="submit" value="Відправити" />
    </form>

    <a href="blog.php">Повернення на головну сторінку сайту</a>
    </body>
    </html>

    <?php
    // Підключення до сервера
    require_once ("connections/MySiteDB.php");
    //Вибір БД
    $select_db = mysqli_select_db ($link, $db);

    //Отримання даних з форми
    $title = $_POST['title'];
    $created = $_POST['created'];
    $article = $_POST[‘article’];

    if (($title)&&($created)&&($article))
        {
        //Формування запиту
        $query = "INSERT INTO notes (title, created, article) VALUES ('$title',
    '$created', '$article’)";
        //Реалізація запиту
        $result = mysqli_query ($link, $query);
        }
    ?>
```

Вправа 3: Сторінка для редагування нотаток

У цій вправі необхідно створити сторінку **editnote.php**, додати назву та пояснювальний текст. Перехід на дану сторінку буде здійснюватися зі сторінки **comments.php** (так як на початку цієї сторінки виводиться текст нотатки, що коментується).

- Відкрийте сторінку **comments.php**. Створіть між текстом коментованої нотатки, та повторюваною областю коментарів порожній абзац і введіть текст «*Змінити нотатку*». Зробіть її гіперпосиланням для переходу на сторінку **editnote.php**.

- Гіперпосилання на **editnote.php**. Для передачі інформації на сторінку **editnote.php** про те, яка саме нотатка модифікується (нотатка з яким ***id***), необхідно передати ідентифікатор нотатки зі сторінки **comments.php** в рядку URL-адреси через гіперпосилання.

- При його одержанні на сторінці **editnote.php** використовується метод GET (принцип роботи аналогічний тому, що був використаний при передачі ідентифікатора нотатки зі сторінки **blog.php** на сторінку **comments.php**), *див. рис. 5.1*:

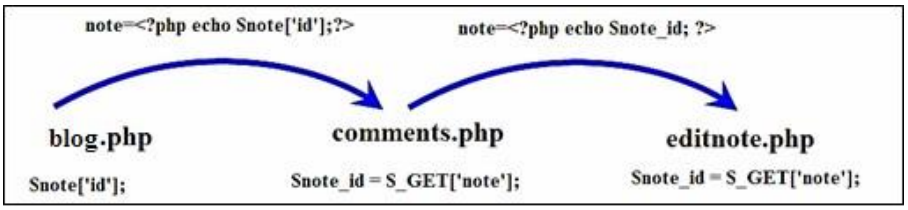

Рис.5.1. Схема обміну даними методом GET

Доповніть гіперпосилання зі сторінки **comments.php** на сторінку **editnote.php**:

*<a href="editnote.php?note=<?php echo $note_id; ?>">Виправити нотатку *

- Робота зі сторінкою **editnote.php**

  - На сторінці **editnote.php** створіть html-форму з назвою «***editnote***», метод обробки даних – POST.
  - На формі розташуйте два поля: одне (типу *text*) для зміни заголовка нотатки – з назвою «***title***», друге (*textarea*) для зміни самої нотатки – з назвою «***article***». Додайте параметри розміру елементів форми.
  - Також помістіть на поле кнопку відправки з назвою «***submit***».
  - Далі необхідно створити php-скрипт для обробки даних форми. Цей скрипт повинен виконувати наступне:

8. Відображати редаговану нотатку в полях форми (тобто розміщати дані з бази в поля форми);
9. Отримувати змінені дані з форми;
10. Передавати змінені дані в таблицю.

- Заповнення полів форми

  - Введіть змінну **$note_id**, яка отримає в якості значення ідентифікатор оброблюваної нотатки. Це значення вона повинна отримати через масив $_GET.
  - Реалізуйте з'єднання з сервером.
  - Виберіть базу даних.
  - Далі необхідно сформувати запит на отримання нотатки з обраним **id** з бази даних, для розміщення її в полях форми. Запит реалізується за допомогою оператора SELECT, умовою запиту має бути **id** вибраної

нотатки.

- Реалізуйте сформований запит.
- За допомогою функції ***mysqli_fetch_array()*** помістіть результат виконання запиту (тобто отриманий рядок) в масив.

Варіант реалізації коду:

```
<?php
//отримання ідентифікатора
$note_id = $_GET['note'];

//З'єднання з сервером
require_once ("connections/MySiteDB.php");

//Вибір БД
$select_db = mysqli_select_db ($link, $db);

//Запит до БД на отримання рядка, що містить нотатку з обраним id
$query = "SELECT * FROM notes WHERE id = $note_id";

//Реалізація запиту до БД
$result = mysqli_query ($link, $query);

//Розміщення вибраного рядка в масив
$edit_note = mysqli_fetch_array ($result);
?>
```

- Необхідно, щоб записи отриманої нотатки відображалися у відповідних полях форми. Для цього:

6.1. Додайте на html-форму приховане поле з назвою note (воно буде містити id нотатки).

6.2. В html-формі задаємо значення value для всіх елементів з масиву:

```
<!-- $edit_note - це назва масиву, в який поміщається результат виконання функції mysqli_fetch_array(); -->

<form id="editnote" name="editnote" method="post" action="">
<label for="title">Заголовок нотатки</label>
<input type="text" name="title" id="title"
        value = "<?php echo $edit_note['title'];?>" />
<label for="article">Текст нотатки </label>
<textarea  name="article" id=" article">
<?php echo $edit_note[article’];?></textarea>
<input type="hidden" name = "note" id = "note"
        value="<?php echo $edit_note['id']?>" />
```

```
<input type="submit" name="submit" id="submit" value="Змінити" />
</form>
```

- Отримання даних з форми після зміни. Принцип реалізації схожий на додавання нової нотатки:

  - Отримайте з форми змінені дані за допомогою методу $_POST;
  - Передайте дані в таблицю за допомогою SQL-запиту. Різниця полягає лише в SQL-запиті - при додаванні використовується INSERT, а при оновленні UPDATE.

Оператор UPDATE оновлює поля таблиці згідно з їх новими значеннями в рядках. Синтаксис запиту на оновлення:

UPDATE tblName SET fieldName1 = expr1, fieldName2 = expr2, ... , fieldName N = expr N WHERE ...

де *tblName* – назва таблиці, *fieldName = expr* - вказується, які саме поля треба змінити і якими мають бути їх нові значення.

Варіант коду отримання і передачі даних з форми

```
<?php
// Оновлення даних
// Отримання оновлених значень з форми
$title = $_POST['title'];
$article = $_POST[‘article’];

//Створення запиту на оновлення

$update_query = "UPDATE notes SET title = '$title', article = '$article’
                                WHERE id = $note_id";
//Реалізація запиту на оновлення
$update_result = mysqli_query ($link, $update_query);
?>
```

- Перевірте коректність роботи скриптів.

- Створіть гіперпосилання для повернення на сторінку коментарів.

Варіант повної реалізації editnote.php коду

```
<?php
$note_id = $_GET['note'];
require_once ("connections/MySiteDB.php");
$select_db = mysqli_select_db ($link, $db);

$query = "SELECT * FROM notes WHERE id = $note_id";
$result = mysqli_query ($link, $query);
$edit_note = mysqli_fetch_array ($result);
```

```
?>

<html>
<body>
<p>Сторінка редагування нотатки </p>

<form id="editnote" name="editnote" method="post" >
<label for="title">Заголовок нотатки</label>
<input type="text" name="title" id="title"
                    value = "<?php echo $edit_note['title'];?>" />
<label for=" article">Текст нотатки </label>
<textarea  name=" article" id=" article">
                    <?php echo $edit_note[‘article’];?></textarea>
  <input type="hidden" name = "note" id = "note"
                    value="<?php echo $edit_note['id']?>" />
<input type="submit" name="submit" id="submit" value="Змінити" />
</form>

<a href="blog.php">Повернутися на головну сторінку сайту</a>
</body>
</html>


<?php
$title = $_POST['title'];
$article = $_POST[‘article’];
$update_query = "UPDATE notes SET title = '$title', article = '$article’
WHERE id = $note_id";
$update_result = mysqli_query ($link, $update_query);
?>
```

Вправа 4: Створення сторінки видалення нотаток

Самостійно створіть сторінку для видалення нотатки deletenote.php. Перехід на цю сторінку також повинен здійснюватися зі сторінки comments.php.

Для реалізації видалення запису з БД використовується SQL- оператор DELETE.

Синтаксис оператора DELETE:

DELETE FROM tblName WHERE …

де tblName – назва таблиці.

Не забудьте реалізувати видалення коментарів до записів, що видаляються.

Лабораторна робота №6: Робота з нотатками

Вправа 1: Робота зі сторінкою blog.php

1. Необхідно зробити так, щоб остання додана замітка відображалася в самому верху списку заміток. Для цього відредагуйте код SQL-запиту до БД таким чином, щоб дані передавалися в необхідному порядку. З цією метою використовуються наступний синтаксис:

SELECT *fieldName* FROM t*blName* ORDER BY *fieldName order fieldName*
– назва поля (полів) таблиці,
t*blName* – назва таблиці – джерела,
order – порядок проходження записів. Він може бути **ASC** - по зростанню, **DESC** - за спаданням, **RAND** - у випадковому порядку.

Наприклад:

*SELECT * FROM table ORDER BY name ASC*

тобто необхідно вибрати все поля з таблиці ***table*** *і розташувати їх у порядку зростання значень поля* ***name*** *(тобто в алфавітному порядку, якщо поле строкового типу).*

2. Необхідно на сторінці deletenote.php зробити посилання повернення на сторінку коментарів. При цьому необхідно враховувати, що при переході за простим посиланням (без додаткових параметрів) нотатка все одно видалиться, тому необхідно в гіперпосиланні «повернути» сторінці коментарів отримані від неї аргументи GET, щоб вона знову вивела ту ж саму нотатку.

Варіант реалізації коду

```
<?php
$note_id = $_GET['note'];
require_once ("connections/MySiteDB.php");
$select_db = mysqli_select_db ($link, $db);
$query = "SELECT * FROM notes WHERE id = $note_id";
$result = mysqli_query ($link, $query);
$delete_note = mysqli_fetch_array ($result);
?>

<html>
<body>
<p>Сторінка редагування нотатки </p>
<form id="editnote" name="editnote" method="post" >
<label for="title">Заголовок нотатки</label>
<input type="text" name="title" id="title"
                    value = "<?php echo $delete_note['title'];?>" />
<label for="article">Текст нотатки </label>
```

```
<input type="text" name=" article" id=" article"
                    value = "<?php echo $delete_note[‘article’];?>" />
<input type="hidden" name = "note" id = "note"
                     value="<?php echo $delete_note['id']?>" />
<input type="hidden" name = "MM_update" value="editnote" />
<input type="submit" name="submit" id="submit" value="Видалити" />
</form>

<?php
$submit = $_POST['submit']; if
($submit)
        {
        $delete_query = "DELETE FROM notes WHERE id = $note_id";
        $delete_result = mysqli_query ($link, $delete_query);
        }
?>
</body>
</html>
```

Вправа 2: Робота з коментарями до нотаток

Самостійно створіть сторінку для додавання коментарів до нотаток, а також сторінку для видалення коментарів.

Лабораторна робота №7: Сторінка статистики inform.php

В ході виконання лабораторної роботи буде організовано роботу і виведені на сторінку статистики наступні дані web-сайту:

1. Скільки всього було зроблено записів у блозі;
2. Скільки коментарів було додано;
3. Скільки записів було зроблено за останній місяць;
4. Скільки коментарів було залишено за останній місяць;
5. Яка нотатка була зроблена останньої;
6. Яку нотатку найбільше коментували.

Вправа 1: Загальна кількість нотаток і загальна кількість коментарів

1. Встановіть підключення до сервера і виберіть базу даних.
2. За допомогою SQL-запиту необхідно обчислити загальну кількість нотаток в блозі. Для цього використовується SQL-функція COUNT ().

Ця функція повертає кількість рядків, які відповідають певним критеріям. Синтаксис функції:

SELECT *COUNT (fieldName)* ***FROM*** *tblName*

Ця функція є **агрегатною**, тобто дозволяє виконувати різні дії відразу над багатьма записами.

Варіант реалізації коду

```
//Обчислення кількості нотаток
$query_allnotes = "SELECT COUNT(id) AS allnotes FROM notes";
$allnotes = mysqli_query ($link, $query_allnotes) or die (mysqli_error());
//mysqli_error()повертає рядок помилки останньої операції з MySQL
$row_allnotes = mysqli_fetch_assoc ($allnotes);
$allnotes_num = $row_allnotes['allnotes'];
mysqli_free_result ($allnotes);
//mysqli_free_result() звільняє пам'ять від результату запиту
```

3. Аналогічним чином реалізуйте підрахунок загальної кількості коментарів.

Вправа 2: Підрахунок кількості нотаток і коментарів за останній місяць

В ході виконання цієї вправи необхідно реалізувати наступний алгоритм роботи:

- Обчислити початкову і кінцеву дати поточного місяця;
- Підставити результати цих обчислень в умову фільтрації SQL-запиту.

1. Робота з датою

//Функція getdate() повертає масив, що містить інформацію про різні складові поточної дати (щоб далі працювати з ними "по частинах"). У масив поміщаються: секунди, хвилини, години, порядковий номер дня, порядковий номер місяця, порядковий номер року, назва дня тижня, назва місяця, кількість секунд з початку епохи Unix.

$date_array = getdate();

//Обчислення початкової дати поточного місяця

//Функція mktime () повертає об'єднане значення часу. Аргументи: к-ть годин, хвилин, секунд, № місяця, число, рік.

$begin_date = date ("Y-m-d", mktime(0,0,0, $date_array['mon'],1, $date_array['year']));

//Оскільки час в даному випадку не потрібен - поставлені нулі.
//Повернене функцією mktime () значення приведене до сприймаємого MySQL параметру дати "Y-m-d".

//Обчислення кінцевої дати поточного місяця
$end_date = date ("Y-m-d", mktime(0,0,0, $date_array['mon'] + 1,0, $date_array['year']));

//Тут все аналогічно, крім того, що ми вводимо число місяця, рівне нулю; на підставі цього функція mktime (), зустрівши дату з нульовим числом, поверне останнє число попереднього місяця (28, 29, 30 або 31).

2. Запит на отримання нотаток за останній місяць

```
$query_lmnotes = "SELECT COUNT(id) AS lmnotes FROM notes
                  WHERE created>='$begin_date' AND created<='$end_date'";
$lmnotes = mysqli_query ($link, $query_lmnotes)or die (mysqli_error());
$row_lmnotes = mysqli_fetch_assoc ($lmnotes);
$lmnotes_num = $row_lmnotes['lmnotes'];
mysqli_free_result ($lmnotes);

//$begin_date – перше число місяця,
//$end_date – останнє число місяця.
```

3. Аналогічним чином обчисліть кількість коментарів за останній місяць.

Вправа 3: Остання додана нотатка

7. Для виведення останньої доданої нотатки необхідно використовувати оператор LIMIT в конструкції SELECT.

8. Вираз LIMIT використовується для обмеження кількості рядків,

повернутих командою SELECT. LIMIT приймає один або два числових аргументи. Ці аргументи повинні бути цілочисельними константами. Якщо задані два аргументи, то перший вказує на початок першого повернутого рядка, а другий задає максимальну кількість повернутих рядків. При цьому зміщення початкового рядка дорівнює 0, а не 1 (тому що перший елемент масиву рядків має індекс 0).

Наприклад:

SELECT * FROM table LIMIT 5,10; // повертає рядки 6-15

Варіант реалізації коду

```
// Остання додана нотатка
// Таблиця notes сортується за датою публікації нотатки за спаданням, а потім з неї береться тільки найперший запис (LIMIT 0,1) - "починаючи з нульовою запису вибрати один запис"

$query_last_note = "SELECT id, title FROM notes
                                    ORDER BY created DESC LIMIT 0,1";
$lastnote = mysqli_query ($link, $query_last_note) or die (mysqli_error());
$row_lastnote = mysqli_fetch_assoc ($lastnote);
mysqli_free_result ($lastnote);
```

Вправа 4: Найбільш коментована замітка

В ході виконання цієї вправи необхідно реалізувати наступний алгоритм роботи:

9. Зв'язати таблиці notes і comments за полями id і art_id відповідно, щоб потім обчислити кількість коментарів для кожної нотатки;
10. Виконати групування таблиці comments за ідентифікатором нотатки.
11. Обчислити кількість коментарів для кожної нотатки.
12. Відсортувати результат за кількістю коментарів для кожної нотатки за зменшенням.
13. Вивести перший запис з отриманого набору записів.

1. Побудова SQL-запиту

```
$query_mcnote = "SELECT notes.id, notes.title FROM comments, notes
                        WHERE comments.art_id=notes.id

                        GROUP BY notes.id
                        ORDER BY COUNT(comments.id) DESC LIMIT 0,1";
```

У тексті запиту:

GROUP BY – оператор групування. Групування - це об'єднання записів в групи по якомусь критерію (критерію групування), який записується відразу після

оператора (в даному випадку групування по полю id таблиці notes).

ORDER BY COUNT (comments.id) DESC LIMIT 0,1 – здійснення сортування по порядку зменшення результатів виконання агрегатної функції COUNT() за id коментарів і висновок першого запису (тобто записи з найбільшою кількістю коментарів).

Реалізуйте даний запит і помістіть його результат в масив.

Вправа 5: Розміщення даних на сторінці

За допомогою php-сценаріїв і оператора echo вивести результати на сторінку сайту. Нижче представлений варіант реалізації коду виведення інформації:

```
<html>
<body>
Зроблено записів - <?php echo $allnotes_num; ?><br>
Залишено коментарів -  <?php echo $allcomments_num; ?><br>
За останній місяць я створив записів - <?php echo
                                        $row_lmnotes['lmnotes'];?><br>
За останній місяць залишено коментарів -  <?php echo
                                $row_lmcomments['lmcomments'];?><br>
Мій останній запис -
                <a href="comments.php?note=<?php echo $row_lastnote['id'];?>">
                                        <?php echo $row_lastnote['title'];?></a><br>
Найбільш обговорюваний запис -
                <a href="comments.php?note=<?php echo $row_mcnote['id'];?>">
                                        <?php echo $row_mcnote['title'];?>
</a><br><br>

<p><a href="blog.php">Повернення на головну сторінку сайту </a></p>
</body>
</html>
```

У представленому коді:

$allnotes_num, $allcomments_num, $row_imnotes, $row_lmcomments, $row_lastnote, $row_mcnote – масиви, в які поміщаються результати виконання функцій mysqli_fetch_array (), що викликаються в раніше створеному коді для отримання і зберігання відповідних даних.

Нижче представлений можливий варіант реалізації всього коду сторінки inform.php (без табличній html-структури).

```
<?php require_once ("connections/MySiteDB.php");?>
<?php
mysqli_select_db ($link, $db);

//Обчислення кількості нотаток
$query_allnotes = "SELECT COUNT(id) AS allnotes FROM notes";
```

```
$allnotes = mysqli_query ($link, $query_allnotes) or die (mysqli_error());
$row_allnotes = mysqli_fetch_assoc ($allnotes);
$allnotes_num = $row_allnotes['allnotes'];
mysqli_free_result ($allnotes);

 //Обчислення кількості коментарів
$query_allcomments = "SELECT COUNT(id) AS allcomments FROM comments";
$allcomments = mysqli_query ($link, $query_allcomments) or die
(mysqli_error());
$row_allcomments = mysqli_fetch_assoc ($allcomments);
$allcomments_num = $row_allcomments['allcomments'];
mysqli_free_result ($allcomments);

//Робота з датою
$date_array = getdate();
$begin_date = date ("Y-m-d", mktime(0,0,0, $date_array['mon'],1,
$date_array['year']));
$end_date = date ("Y-m-d", mktime(0,0,0, $date_array['mon'] + 1,0,
$date_array['year']));

//Нотатки за останній місяць
$query_lmnotes = "SELECT COUNT(id) AS lmnotes FROM notes
                  WHERE created>='$begin_date' AND created<='$end_date'";
$lmnotes = mysqli_query ($link, $query_lmnotes)or die (mysqli_error());
$row_lmnotes = mysqli_fetch_assoc ($lmnotes);
$lmnotes_num = $row_lmnotes['lmnotes'];


mysqli_free_result ($lmnotes);

//Коментарі за останній місяць
$query_lmcomments = "SELECT COUNT(id) AS lmcomments FROM comments
                  WHERE created >= '$begin_date' AND created <=
'$end_date'";
$lmcomments = mysqli_query ($link, $query_lmcomments)or die (mysqli_error());
$row_lmcomments = mysqli_fetch_assoc ($lmcomments);
$lmcomments_num = $row_lmcomments['lmcomments'];
mysqli_free_result ($lmcomments);

//Остання додана нотатка
$query_last_note = "SELECT id, title FROM notes
                                    ORDER BY created DESC LIMIT 0,1";
$lastnote = mysqli_query ($link, $query_last_note) or die (mysqli_error());
$row_lastnote = mysqli_fetch_assoc ($lastnote);
mysqli_free_result ($lastnote);

//Найбільш коментована нотатка
$query_mcnote = "SELECT notes.id, notes.title FROM comments, notes
```

```
                    WHERE comments.art_id=notes.id
                    GROUP BY notes.id
                    ORDER BY COUNT(comments.id) DESC LIMIT 0,1";
$mcnote = mysqli_query($link, $query_mcnote)or die (mysqli_error());
$row_mcnote = mysqli_fetch_assoc($mcnote);
mysqli_free_result ($mcnote);
?>

<html>
<body>
Зроблено записів - <?php echo $allnotes_num; ?><br>
Залишено коментарів - <?php echo $allcomments_num; ?><br>
За останній місяць я створив записів - <?php echo
                    $row_lmnotes['lmnotes'];?><br>
За останній місяць залишено коментарів - <?php echo
                    $row_lmcomments['lmcomments'];?><br>
Мій останній запис -
        <a href="comments.php?note=<?php echo $row_lastnote['id'];?>">
                    <?php echo $row_lastnote['title'];?></a><br>
Найбільш обговорюваний запис -
        <a href="comments.php?note=<?php echo $row_mcnote['id'];?>">

</a><br><br>

<?php echo $row_mcnote['title'];?>

<p><a href="blog.php">Повернення на головну сторінку сайту </a></p>
</body>
</html>
```

Лабораторна робота №8: Реалізація пошуку по сайту

В ході виконання цієї лабораторної роботи будуть вивчені основні функції роботи з рядками, а також пошук інформації по web-сайту (за одним і декількома словами пошукового запиту).

Вправа 1: Реалізація пошуку по сайту

В ході виконання цієї вправи з використанням функцій роботи з рядками необхідно реалізувати можливість пошуку за ключовим словом нотатки на головній сторінці сайту (можливі два варіанти реалізації: пошук за одним словом і пошук за фразою).

8. Пошук за одним ключовим словом: Для реалізації пошуку за одним словом можна використовувати оператор LIKE і замінники символів %.

Варіант реалізації коду:

```
// Пошук за одним словом
  $user_search  =  $_GET['usersearch'];
  if (!empty($user_search))
        {
        $query_usersearch = "SELECT * FROM notes
              WHERE  title  LIKE  '%$user_search%'
              OR article LIKE '%$user_search%'";
        $result_usersearch = mysqli_query($link, $query_usersearch);
        while ($array_usersearch = mysqli_fetch_array($result_usersearch))
              {
              echo $array_usersearch['id']; echo
              $array_usersearch['title'];      echo
              $array_usersearch['article'];
              }

    }
```

9. Реалізація пошуку за фразою:

1. Фразу потрібно розбити на окремі слова (підрядки) і розмістити в масив підрядків за допомогою функції ***explode();***

```
$search_query = “SELECT * FROM tableName”
$where_clause = ‘ ‘;  // умова пошуку
$user_search = $_GET['usersearch']; // отримуємо дані з поля пошуку
$search_words = explode(' ', $user_search);

foreach($search_words as $word)
   {
      // Формуємо умову пошуку
      $where_clause .= ” fieldName LIKE ‘%$word%’ OR “
   }
```

```
if (!empty($$where_clause))
    {
        $search_query .= " WHERE $$where_clause ";
    }
```

2. Для того, щоб в кінці рядка запиту не було оператора OR, можна використовувати функцію implode (), що створює рядок з масиву підрядків, переданого їй в якості аргументу.

Представлений раніше код можна змінити наступним чином:

```
$search_query = "SELECT * FROM tableName"
$where_list = array();
$user_search = $_GET['usersearch'];
$search_words = explode(' ', $user_search);

foreach($search_words as $word)
   {
      //В кінець масиву додається новий елемент
      $where_list[] = "article LIKE '%$word%'";
   }
$where_clause = implode (' OR ', $where_list);

    if (!empty($where_clause))
       {
            $query_usersearch .=" WHERE $where_clause";
       }
```

Варіант реалізації коду:

```
//Пошук за фразою (за змістом нотатки)

$user_search = $_GET['usersearch'];
$where_list = array();
$query_usersearch = "SELECT * FROM notes";
$search_words = explode(' ', $user_search);

foreach($search_words as $word)
       {
      $where_list[] = " article LIKE '%$word%'";
       }
$where_clause = implode (' OR ', $where_list);
if (!empty($where_clause))
       {
            $query_usersearch .=" WHERE $where_clause";
       }
```

```
$res_query = mysqli_query($link, $query_usersearch);
        while ($res_array = mysqli_fetch_array($res_query))
                {
                echo $res_array['id'], "<br>";
                echo $res_array[‘article’], "<br>", "<hr>", "<br>";
}
```

Вправа 2: Обробка рядку пошуку

Рядок пошуку повинен мати декілька слів, розділених одним пробілом. Але потрібно враховувати, що користувач може вводити слова пошукового запиту через кому (наприклад, «замітка, моя, нова»). Такий рядок необхідно перед передачею до запиту до бази даних обробити та привести до необхідного виду. Мінімальна обробка здійснюється у два етапи:

0. Заміна ком на пробіли;

1. Видалення зайвих пробілів між словами рядка.

1. Заміна ком на пробіли здійснюється за допомогою функції ***str_replace().*** Обов’язковими є три аргументи: що замінити, чим замінити, де замінити. Отже, в даному випадку виклик функції буде виглядати наступним чином:

```
str_replace (‘ , ’, ‘ ‘, $user_search);
```

2. Видалення зайвих пробілів між словами рядка пошукового запиту: в тому випадку, коли коми були замінені на пробіли, з’явилися надлишкові пробіли (тобто більше одного) між словами запиту. Якщо їх не видалити, то в запиті вони будуть розглядатися як пусті елементи масиву, на основі якого формується запит до бази даних. Отже, при такому запиті будуть видаватися всі записи бази даних. Для видалення пустих елементів масиву можна зробити наступне:
   - Створити новий масив, в якому будуть зберігатися тільки дійсні (непусті) критерії пошуку. На основі цього масиву буде створюватися запит до бази даних.
   - Для створення такого масиву можна пройти в циклі *foreach* всі елементи вже існуючого масиву, використовуючи умовну конструкцію *if*, знайти всі непусті елементи та скопіювати їх у новий масив.

1. У новий масив ***$final_search_words*** додаються непусті елементи вже існуючого масиву ***$search_words***.

```
//Вилучення критеріїв пошуку в масив
//Заміна ком на пробіли
$clean_search = str_replace(',', ' ', $user_search);
$search_words = explode(' ', $user_search);

//Створюємо ще один масив з кінцевими результатами
$final_search_words = array();

//Проходимо в циклі по кожному елементу масива $search_words.
//Кожен непустий елемент додаємо в масив
//з назвою $final_search_words
```

```
if (count($search_words) > 0)
{
        foreach($search_words as $word)
        {
                if (!empty($word))
                {
                        $final_search_words[] = $word;
                }
        }
}
```

2. Далі реалізація роботи з масивом така ж, різниця тільки в масиві, що ми використовуємо (будемо працювати з новим масивом **$final_search_words**).

```
foreach ($final_search_words as $word)
{
        $where_list[] = " article LIKE '%$word%'";
}
$where_clause = implode (' OR ', $where_list); if
(!empty($where_clause))
{
        $query_usersearch .=" WHERE $where_clause";
}
```

Варіант реалізації коду:

```
<?php
//Пошук за фразою (за змістом замітки)
$user_search = $_GET['usersearch'];
$where_list = array();
$query_usersearch = "SELECT * FROM notes";
$clean_search = str_replace(',', ' ', $user_search);
$search_words = explode(' ', $user_search);

//Створюємо ще один масив з кінцевими результатами
$final_search_words = array();

//Проходимо в циклі по кожному елементу масива $search_words.
//Кожен непустий елемент додаємо в масив $final_search_words
if (count($search_words) > 0)
{
        foreach($search_words as $word)
        {
                if (!empty($word))
                {
                        $final_search_words[] = $word;
                }
        }
}
//Оброблюємо масив $final_search_words
foreach ($final_search_words as $word)
{
        $where_list[] = " article LIKE '%$word%'";
}
```

```
$where_clause = implode (' OR ', $where_list);
if (!empty($where_clause))
{
        $query_usersearch .=" WHERE $where_clause";
}
$res_query = mysqli_query($link, $query_usersearch);
while ($res_array = mysqli_fetch_array($res_query))
{
        echo $res_array['id'], "<br>";
        echo $res_array[‘article’], "<br>", "<hr>", "<br>";
}
?>
```

Лабораторна робота №9: Передача файлів на сервер

У даній лабораторній роботі будуть показані основні можливості PHP для реалізації передачі файлів на сервер.

Вправа 1: Вивід списку файлів

1. Створіть на локальному вузлі (у нашому випадку в папці C:\WebServers\home\localhost\www\) папки **photo** та **files** для розміщення зображень та файлів відповідно. Помістіть в ці папки декілька зображень та документів.

2. Створіть сторінку **photo.php**, розмістіть на ній пояснювальний текст, нижче цього тексту – дві горизонтальні лінії (між цими лініями буде виводитися список доступних на web-вузлі зображень та посилань на ці зображення), див. *рис. 9.1*.

Рис. 9.1. Схема сторінки photo.php

3. Для виводу списку файлів, які є на сервері, необхідно створити два сценарії:

11. перший буде формувати список файлів із зображеннями,
12. другий – виводити цей список на сторінку.

1. Формування списку файлів. Для формування списку файлів необхідно отримати шлях до цільової папки, де зберігаються потрібні нам файли, а потім створити масив з необхідною інформацією про файли (імена, шляхи до них);

Варіант реалізації коду:

```
<?php

//Отримуємо повний шлях до папки, в якій зберігаються графічні файли
$image_dir_path = $_SERVER['DOCUMENT_ROOT'] . "/photo";
```

```
//Запускаємо перегляд папки. Функція opendir() повертає
// ідентифікатор папки
$image_dir_id = opendir($image_dir_path);

//$array_files – масив, у який будуть розміщені всі знайдені файли
$array_files = null;

//Службова змінна для розрахунку індекса наступного
//елемента масиву $array_files

$i = 0;

//Запускаємо цикл перегляду
//Функція readdir() повертає повний шлях до наступного файлу
//в папці, ідентифікатор якої був повернутий функцією opendir() та переданий як параметр.
//$path_to_file отримує повний шлях до файлу для подальшої обробки. Якщо в папці немає
непереглянутих файлів – повертається логічне значення false
while(($path_to_file = readdir($image_dir_id)) !== false)
{
        //Крапки означають вкладені файли: одна крапка – поточна папка, дві крапки – папка, в
        яку вкладена поточна папка.
        if(($path_to_file !=".") && ($path_to_file !=".."))
        {
                //Розміщуємо ім'я знайденого файлу в масив $array_files. Функція basename() дозволяє
                отримати ім'я файлу з повного шляху до нього.
                $array_files[$i] = basename($path_to_file);
                $i++;
        }
}
//closedir() видаляє з пам'яті переданий їй ідентифікатор папки,
//таким чином завершуючи перегляд.
closedir($image_dir_id);
?>
```

2. Вивід списку файлів на сторінку

- Знайдіть код, який створює дві горизонтальні лінії. Створіть наступний сценарій виводу списку файлів:

```
<?php
//Отримуємо кількість елементів масиву $array_files, тобто кількість
//знайдених файлів.
$array_files_count = count($array_files);
if ($array_files_count)
{
        <?php

        ?>
        <hr />


        sort($array_files);
        for ($i=0; $i<$array_files_count; $i++)
        {
        //Виводимо імена файлів з масиву на сторінку
        ?>
```

```
        <p><a href="/photo/<?php echo $array_files[$i]; ?>" target="_blank">
                <?php echo $array_files[$i]; ?></a></p>
        <?php
        <?php
        }
 ?>
}
?>
<hr />
```

- Створіть гіперпосилання для повернення на головну сторінку та гіперпосилання з головної сторінки на сторінку **photo.php** («Фото» в наборі гіперпосилань).

4. Перевірте роботу сценаріїв.

Вправа 2: Відправлення файлів на сервер

Відправлення файлів на web-сайт складається з двох етапів:

13. Створення необхідної форми для відправлення файлів на сторінці сайту;
14. Написання сценарію PHP для отримання відправленого користувачем файлу.

1. Створення форми. Стандарт HTML передбачає так зване ***поле вибору файлу***, в яке відвідувач повинен буде ввести ім'я файлу, який відправляється. Від звичайного поля вводу воно відрізняється тим, що дозволяє працювати з вікном відкриття файлів Windows, а також воно само відправляє серверній програмі не введене ім'я файлу, а сам файл. Поле вибору файлу створюється за допомогою тегу INPUT, атрибут *type* якого має значення «file». Також в форму з даним полем потрібно додати скрите поле, що задає максимальний розмір файлу, який відправляється, з іменем MAX_FILE_SIZE. Значення цього поля (тобто максимально можливий розмір файлу) зазначається в байтах. Форма, з якої буде відправлятися файл, повинна кодувати дані по методу multipart/form-data та передавати дані тільки методом POST.

```
<!-- Форма для відправлення файлу на сервер -->
<form name = "file_upload" action="photo.php"
                        enctype="multipart/form-data" method="post">
<input type="hidden" name="MAX_FILE_SIZE" value="65536" />
<input type="file" name="file_upload" />
<input type="submit" name="submit" value="Додати" />
</form>
```

   В даному випадку максимальний розмір файлу дорівнює 65536 байтам, тобто 64 Кбайти. Даний розмір при необхідності може бути збільшений.

2. Отримання відправленого файлу. Усі отримані файли розміщуються інтерпретатором PHP в особливу службову папку, яка не є частиною сайту (так звана «буферна» папка). Дані про всі прийняті та поміщені у буферну папку файли зберігаються у вбудованому масиві PHP $_FILES. Кожен елемент масиву відповідає прийнятому файлу та представляє собою вкладений масив з різними відомостями про файл. На початку сторінки розмістіть наступний код:

```
<?php
//Сценарій відправки файлу на сервер
//Перевіряємо, чи було виконано відправлення файлу. Далі реалізуємо сценарій.
if (isset($_POST["MAX_FILE_SIZE"]))
{
        $tmp_file_name = $_FILES["file_upload"]["tmp_name"];
        $dest_file_name = $_SERVER['DOCUMENT_ROOT'].
                        "/photo/".$_FILES["file_upload"]["name"];
        move_uploaded_file($tmp_file_name, $dest_file_name);
}
?>
```

Функція ***move_uploaded_file (string filename, string destination)*** перевіряє, чи був файл *filename* завантажений на сервер (переданий по протоколу HTTP POST). Якщо файл дійсно був завантажений, він буде переміщений у місце, вказане в аргументі *destination*.

Вправа 3: Видалення файлу з сервера

Для видалення файлу потрібно вибрати необхідний файл. Це можна зробити за допомогою форми, у якій знаходиться список файлів для видалення із тих, що доступні на сервері. Для заповнення цього списку необхідно використовувати вже створений масив $array_files.

Реалізація видалення проходить в два етапи:

- Створення форми для видалення файлу;
- Створення сценарію для видалення файлу.

1. Створення форми для видалення файлу

```
<!—Форма для видалення файлу з сервера -->
<form name="file_delete" action="photo.php" method="post"
                        enctype="application/x-www-form-urlencoded">
Файл <select name = "file_delete" size="1">
                        <option><option></select>
<input type="submit" name="submit" value="Видалити" />
</form>
```

Парний тег *select* створює список. Пункти списку створюються тегами *option*. Список файлів на сервері необхідно отримати зі створенного раніше масиву $array_files.

```
<!—Форма для видалення файлу з сервера -->
<form name="file_delete" action="photo.php" method="post"
                        enctype="application/x-www-form-urlencoded">
Файл <select name = "file_delete" size="1">
<?php for ($i=0; $i<$array_files_count; $i++) { ?>
<option><?php echo $array_files[$i]; ?></option>
<?php } ?></select>
<input type="submit" name="submit" value="Видалити" />
</form>
```

У доданому до форми циклі створюється стільки пунктів (тегів <option>), скільки елементів є в масиві $array_files.

2. Створення сценарію видалення файлу.

```
<?php
//Сценарій видалення файлу
//Спочатку перевіряємо, чи було ініційовано видалення файлу
if (isset($_POST["file_delete"]))
{

        //Формуємо повне ім’я файлу
        $file_name = $_SERVER['DOCUMENT_ROOT'] . "/photo/".
                                                        $_POST["file_delete"];
        //Функція unlink() видаляє файл
        unlink($file_name);
}
?>
```

3. Збережіть зміни та перевірте коректність роботи.

Вправа 4: Створення сторінки роботи з файлами

Самостійно створіть сторінку для роботи з файлами **files.php**. Вона повністю аналогічна сторінці **photo.php**, окрім того, що:

1. Вона повинна обробляти файли, які знаходяться в папці *files*.
2. Перегляд файлів робити необов’язково (це не зображення), тому не потрібно робити гіперпосилання з імені кожного файлу.
3. Необхідно збільшити максимальний розмір файлу хоча б до 1-го Мбайту.
4. Створіть гіперпосилання, які пов’язують дану сторінку з головною сторінкою.

Лабораторна робота №10: Автоматизація роботи засобами інструментального середовища Adobe Deamweaver. Розмежування доступу до розділів сайту

У ході виконання цієї лабораторної роботи будуть продемонстровані можливості Adobe Dreamweaver для автоматизації створення php-скриптів, які часто використовуються при написанні web-додатків.

Вправа 1: Автоматизація розміщення даних на сторінці

1. Створіть на локальному вузлі (в нашому випадку в папці C:\WebServers\home\localhost\www\) папку Dreamweaver. Створіть файл **main.php** – на ньому розмістіть тільки елементи, вказані на *рис. 10.1*. Цей файл буде зменшеною «копією» файлу **blog.php**.

   Усі файли, створені у цій практичній роботі необхідно також зберегати в папці Dreamweaver.

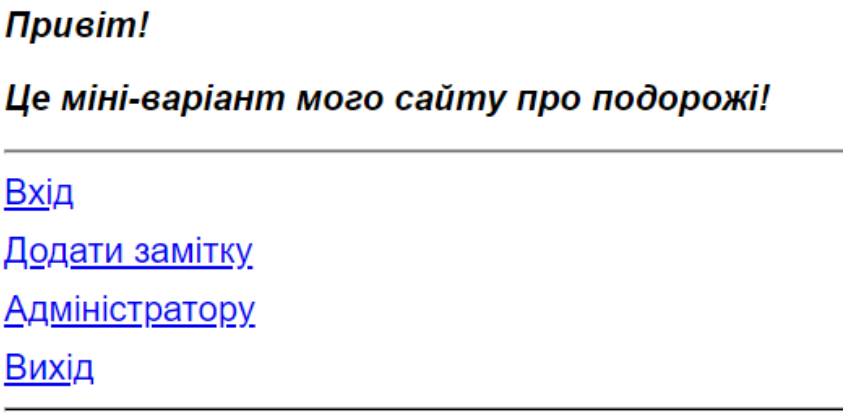

Рис. 10.1. Схема файлу main.php

2. Автоматично створіть ще одне з'єднання з сервером (**Бази даних** – «+» – **З'єднання MySQL**), збережіть усі налаштування **Dreamweaver**, назвіть з'єднання ***Dreamweaver*** *(рис. 10.2)*.

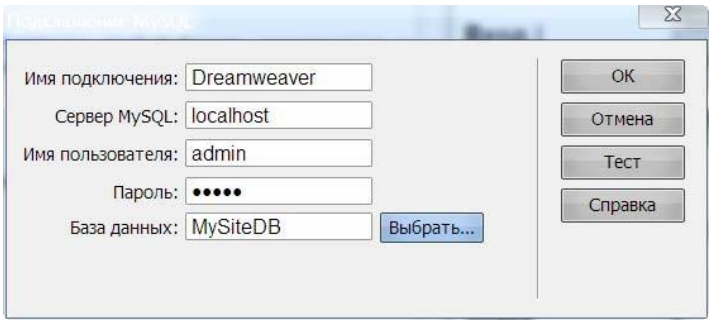

Рис.10.2. Вікно створення з'єднання з сервером

3. Відкрийте файл за кодом з'єднання (**Connections/Dreamweaver.php**). Додайте в кінець скрипта рядки з кодуванням кирилиці:

```
mysql_query("SET NAMES cp1251;" , $Dreamweaver) or die(mysql_error()); mysql_query("SET CHARACTER SET cp1251;", $Dreamweaver) or die(mysql_error());
```

4. У результаті у вас з'явиться з'єднання з БД "*MySiteDB*" під назвою ***Dreamweaver***. Усі з'єднання Dreamweaver автоматично розміщує в одній папці з відповідною назвою.

5. Сторінка **main.php** є аналогом сторінки **blog.php**, але створювати ми її будемо цілком за домопогою автоматизації **Dreamweaver**. Вона також буде мати список заміток блогу. Крім того, ми розмістимо на ній деякі додаткові елементи. Етапи роботи:
   0. Автоматичне створення набору записів (рекордсету), тобто масиву, в якому розміщуються записи з БД.
   1. Автоматичне розміщення записів на сторінці сайту.

Пам'ятайте:більшу частину дій необхідно виконувати в режимі дизайну.

6. Створення набору записів (рекордсету) сторінки **main.php.** Для додавання заміток з БД на сторінку блогу потрібно створити рекордсет (набір записів). Таким чином, записи з БД розміщуються в автоматично створений масив.
   1. Виберіть вкладку **Прив'язки – «+» - Набір записів (Запит).**

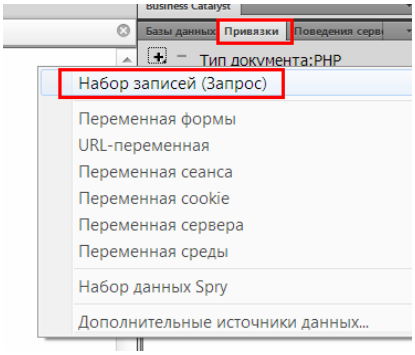

Рис. 10.3. Меню Прив'язки

   2. Відкриється вікно автоматичного створення набору записів. Заповніть поля цього вікна (*рис. 10.4*):

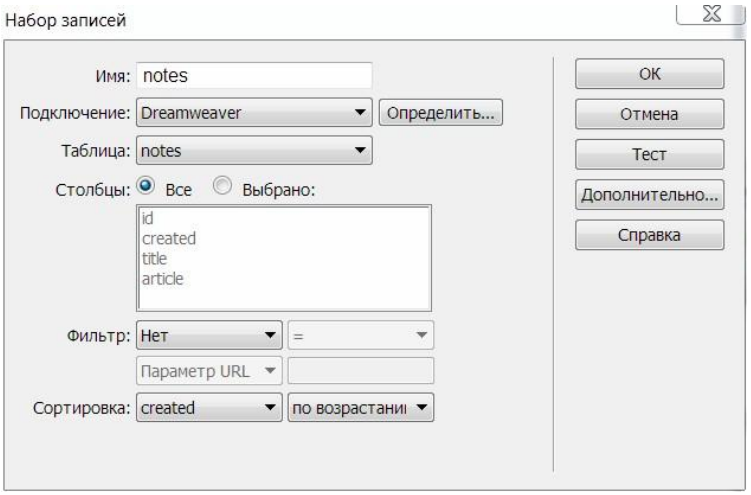

Рис. 10.4. Вікно автоматичного створення набору записів

7. Далі необхідно розмістити замітки на сторінці сайту. ***Пам'ятайте, що автоматичне розміщення елементів на сторінці ЗАВЖДИ проводиться в режимі Дизайну.*** Для цього мишею перетягніть вміст рекордсету з вкладки Прив'язки на сторінку сайту (рис. 10.5).

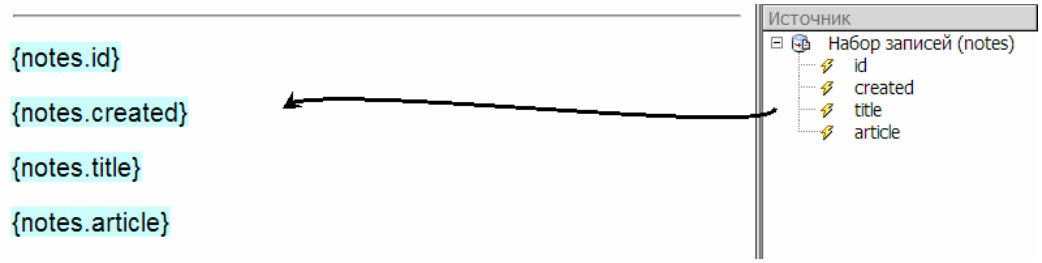

Рис. 10.5. Розміщення вмісту рекордсету на сторінці в режимі дизайну

8. Створення повторюваної області. На даний момент додається тільки перша замітка сайту. Необхідно зробити так, щоб додавалися усі замітки, що є в базі. Для цього необхідно організувати повторювану область:
   8.1. В режимі дизайну виділіть ті елементи, які необхідно повторювати (в нашому випадку – всі елементи рекордсету)
   8.2. Виберіть **Поведінка сервера – «+» - Повторити область**.
   8.3. Встановіть вивід 10 записів на сторінці за один раз.
   8.4. У результаті на сторінці з'явиться повторювана область.

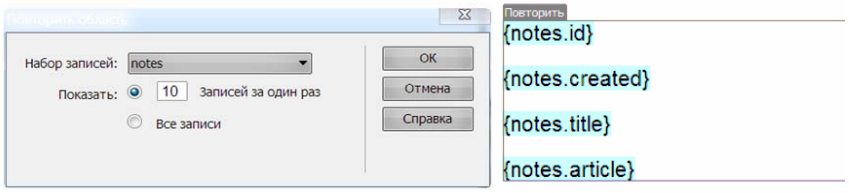

Рис. 10.6. Зовнішній вигляд повторюваної області

Вправа 2: Створення посторінкового навігатора

Раніше була встановлена умова – вивід не більше 10 заміток на сторінку. Якщо заміток в базі більше, потрібно надати можливість переходів між сторінками, що відображають замітки.

1. Додайте до БД більше, ніж 10 заміток.

2. Створіть на сторінці **main.php** абзац з набором навігаційних гіперпосилань (*рис. 10.7*):

Привіт!

Це міні-варіант мого сайту про подорожі!

На головну | Вперед | Назад | На останню

Вхід
Додати замітку
Адміністратору
Вихід

Рис. 10.7. Абзац з набором навігаційних гіперпосилань

3. Для створення гіперпосилань навігатора необхідно виділити слово/фразу (наприклад, «На головну»), вибрати на панелі **Поведінка Сервера** знак «+», підменю **Розбивка набору записів на сторінки** та вибрати команду

«Разбивка набору записів на сторінки».

4. Створіть всі необхідні посилання навігатора. Перевірте коректність їх роботи.
5. Далі необхідно створити на сторінці **main.php** інформацію про загальну кількість замітоk в наборі записів та про кількість замітоk, які відображаються в даний момент на конкретній сторінці. Принцип реалізації аналогічний принципу розбивки по сторінкам:
    6. Розмістіть в нижній части сторінки запис «*Замітки з… по … із…*»
    7. Далі: вкладка **Поведінка сервера – Відобразити лічильник записів**.
    8. Додайте відповідні записи.

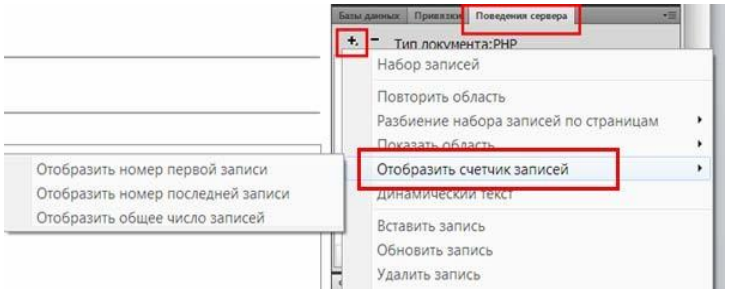

Рис. 10.8. Відображення лічильника записів

Вправа 3: Обмін даними між сторінками

1. Створіть сторінку для виводу коментарів **comm.php**.
2. Створіть посилання між сторінками (пам’ятайте, що ми працюємо з новими сторінками: **main.php** та **comm.php**). Для переходу з головної сторінки на сторінку коментарів посиланням буде заголовок замітки.
3. Самостійно організуйте передавання номеру замітки між сторінками, при цьому для коректної роботи звертайте увагу на те, які змінні Dreamweaver ввів автоматично та використовуйте їх. Для передачі номеру замітки до гіперпосилання на сторінці **main.php** додайте ідентифікатор **note** з відповідним значенням. На сторінці коментарів він буде отриманий автоматично.
4. На сторінці **comm.php** створіть два рекордсети (*рис. 10.9*): один (***notes***) для виводу прокоментованої замітки, другий – з іменем ***comments*** для виводу коментарів до замітки.
5. Зверніть увагу на Фільтр «**Параметр URL**» – у ньому ми автоматично отримуємо ідентифікатор замітки **note**, що був переданий за посиланням.

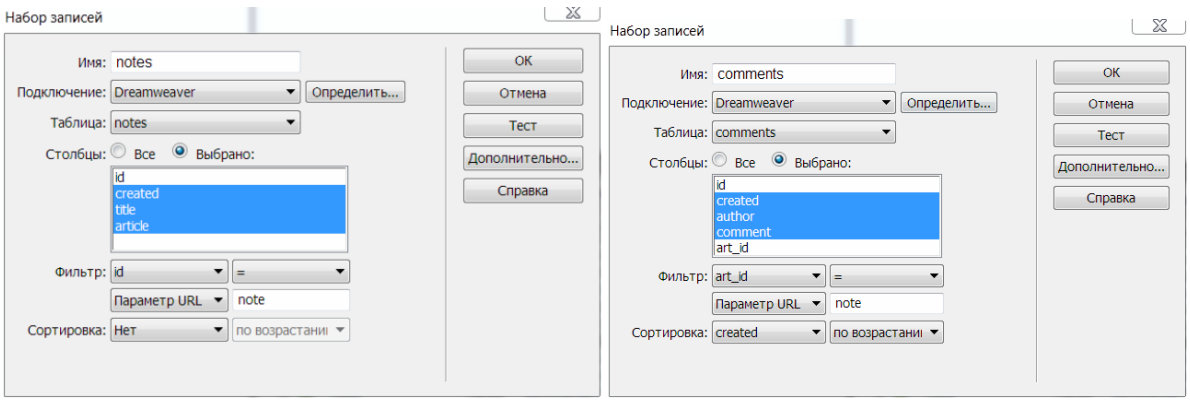

Рис. 10.9. Створення рекордсетів notes та comments

6. Перетягніть на сторінку всі необхідні поля.
7. Створіть для коментарів повторювану область.

Вправа 4: Сторінка додавання замітки

1. Створіть сторінку **addnew.php** для автоматичного додавання користувачем нової замітки.
2. Створіть на сторінці форму (меню **Вставка – Форма**). Ім'я форми – **addnote.**
3. Поставте в форму текстовий курсор. Внизу сторінки з'явиться редактор форм (якщо він не відкривається автоматично, виберіть меню **Вікно – Властивості**). Заповніть форму:

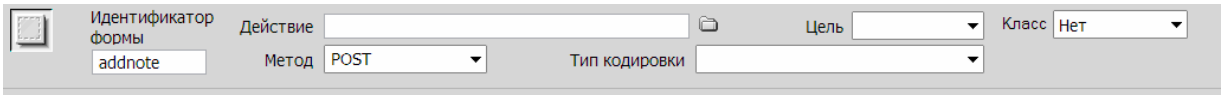

Рис. 10.10. Вікно властивостей форми

4. Додайте до форми текстове поле (**Вставка – Форма – текстове поле**). **Id** та **name** поля – ***title***; надпис – ***Заголовок***.

5. Встановіть властивості текстового поля:

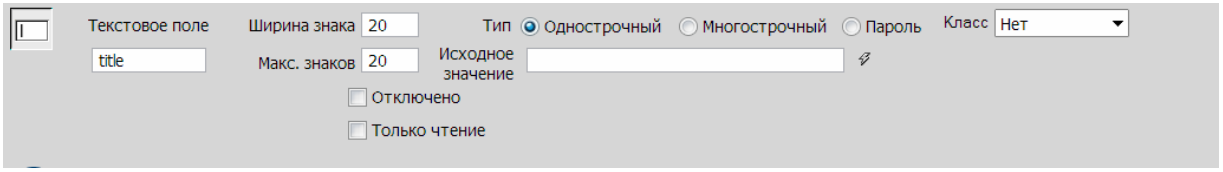

Рис. 10.11. Вікно властивостей текстового поля

6. Аналогічно створіть у формі область з текстом ***Вміст*** з назвою ***article*** з наступними властивостями:

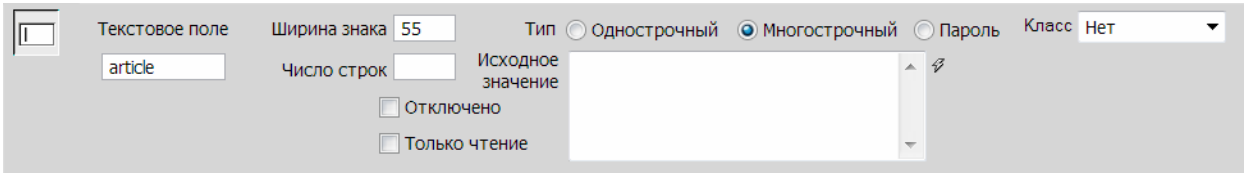

Рис. 10.12. Вікно властивостей області тексту

7. Далі власноручно до форми необхідно додати скрите поле з іменем ***created***, що автоматично отримує дату створення замітки (функція ***date()***).
8. Завершуючи, додайте до форми кнопку **Submit** (**Вставка – Форма** - **Кнопка**).

Рис. 10.13. Форма додавання нової замітки

9. Для автоматичного додавання в БД нової замітки виберіть вкладку **Поведінка сервера** – "+" – **Додати запис**. Заповніть форму, що відкрилася:

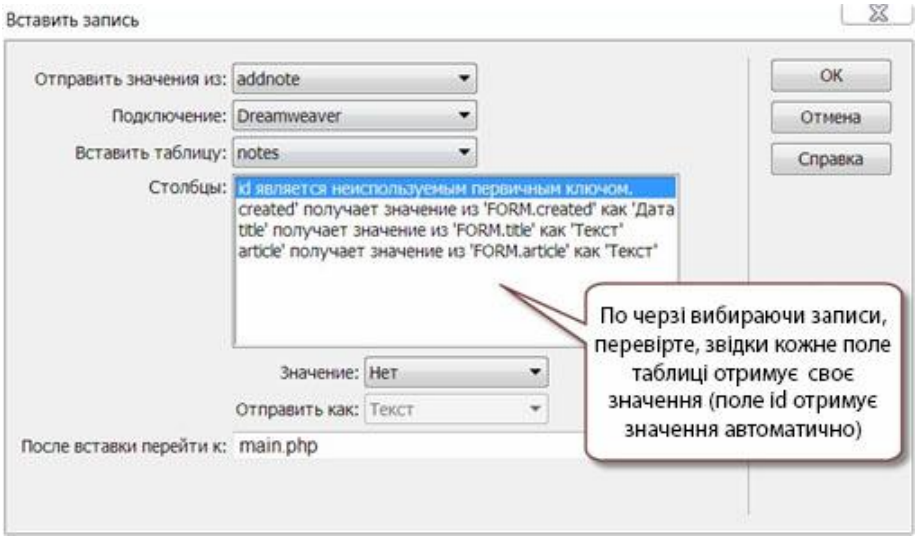

Рис. 10.14. Форма автоматизації додавання записів до бази

10. Перевірте роботу форми (відображення доданих через форму даних в БД "MySiteDB").

11. Створіть гіперпосилання **main.php** на **addnew.php**.

12. Оновлення та видалення записів також автоматично реалізується за допомогою відповідних команд у меню. Власноручно потрібно лише передати ідентифікатор замітки та значення полів, які передаються в форму.

Вправа 5: Розмежування доступу до даних

Для створення розмежування доступу до сторінок сайту необхідно вирішити наступні задачі:

- Створити в БД MySiteDB нову таблицю зі списком користувачів «privileges». Ця таблиця буде мати поля, що будуть зберігати імена користувачів, паролі та відомості щодо прав доступу до різноманітних сторінок сайту.
- Коли користувач буде намагатися переглянути закриті сторінки сайту, запитати у нього ім'я користувача та пароль.
- Якщо ім'я та пароль, введені користувачем, є в БД, потрібно перенаправити користувача на головну сторінку сайту та зробити доступними посилання.
- Якщо ім'я користувача та пароль в БД відсутні, потрібно зробити доступною для користувача тільки головну сторінку сайту; гіперпосилання для нього повинні бути недоступними.
- Дати можливість зареєстрованим користувачам коректно покинути сайт.
- Розподілити всіх зареєстрованих користувачів на дві категорії: users(u), administrators(a). Для адміністраторів (administrators) повинні бути доступними всі сторінки сайту, а для простих користувачів (users) частина сторінок повинна бути закритою для доступу.

1. Створення таблиці списку користувачів. Створіть в БД «MySiteDB» таблицю «privileges», см. *Додаток 3.*

2. Додайте в таблицю один запис, що є адміністратором (наприклад, admin, admin, a).

3. Створіть сторінку входу на сайт **login.php**. На цій сторінці необхідно

створити форму для вводу імені користувача та паролю. Для цього:

4. Створіть форму з іменем ***login***.
5. До створеної форми додайте два текстових поля для вводу та кнопку. Параметри першого поля: ім'я – ***name***, ширина – ***20***, максимальна кількість символів – ***20***. Параметри другого поля: ім'я – ***password***, ширина та максимальна кількість символів – ***20***, тип – ***поле вводу пароля***. Кнопка: ім'я – ***submit***, надпис – «***Увійти***».
6. Далі виберіть **Поведінка сервера** – «+» - **Перевірка справжності користувача – Вхід користувача до системи**.
   Заповніть вікно, що з'явилося після цього:
7. *Отримати дані з форми:* ***login****;*
8. *Поле імені користувача:* ***name****;*
9. *Поле пароля:* ***password****;*
10. *Перевірка за допомогою підключення:* ***Dreamweaver****;*
11. *Таблиця:* ***privileges****;*
12. *Стовпець іменя користувачів:* ***name****;*
13. *Стовпець паролів:* ***password****;*
14. *Якщо увійти вдалося, перейдіть на:* ***main.php***
15. *Якщо увійти не вдалося, перейдіть на:* ***main.php*** *(якщо користувач не пройшов перевірку, він має право на доступ тільки до головної сторінки. Варіант: створити сторінку-«заглушку», на яку в цьому випадку буде переадресований користувач).*
16. *Обмеження доступу на основі:* ***Ім'я користувача, пароль та рівень доступу.***
17. *Отримати рівень доступу з:* ***rights****.*
18. Створіть на сторінці **main.php** гіперпосилання на сторінку **login.php** (в меню всіх гіперпосилань слово «Вхід»).
19. Необхідно обмежити вхід на сторінку **addnew.php** незареєстрованим користувачам.
20. Відкрийте сторінку **addew.php**.

    Поведінка сервера – «+» - Перевірка справжності користувача – Обмеження доступу до сторінки.
21. Заповніть діалогове вікно, що з'явилося:

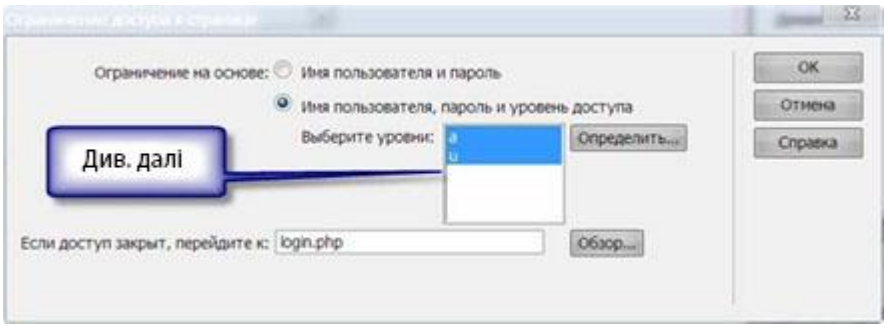

Рис. 10.15. Форма обмеження доступу до сторінки

22. Визначення рівней доступу. Права на доступ «**u**» та «**a**» додайте власноручно, натиснув кнопку на формі «*Визначити…*». Відкриється діалогове вікно визначення рівня доступу (*рис. 10.16)*. Введіть інформацію щодо прав доступу, використовуючи «+» для додавання значень.

Після створення рівней доступу не забудьте у вікні «Обмеження доступу до сторінки» (попереднє вікно) вибрати обидва пункти - «***u***» и «***a***», щоб дати доступ до цієї сторінки як зареєстрованим користувачам, так і адміністраторам.

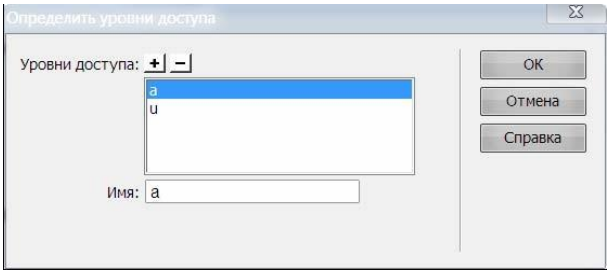

Рис. 10.16. Діалогове вікно визначення рівнеей доступу

23. Створіть сторінку **logout.php** – сторінку виходу з сайту. Створіть на ній пояснювальний текст. Для виходу з сайту користувач повинен буде клацнути розташоване на цій сторінці гіперпосилання (це також дає можливість користувачу передумати та повернутися до роботи з сайтом під своїм логіном на паролем).
24. Створіть на сторінці **logout.php** гіперпосилання «Вихід» та виділіть її. Далі **Поведінка сервера** – «+» - **Перевірка справжності користувача – Вихід користувача з системи**. З'явиться вікно **Log Out User**, у ньому необхідно задати налаштування моменту виходу з сайту:

Рис. 10.17. Вікно організації виходу з системи

На цій же сторінці задайте гіперпосилання повернення на головну сторінку (щоб користувач міг повернутися, не виходячи зі свого облікового запису).

Рис. 10.18. Схема розташування гіперпосилань

25. Створіть на сторінці **main.php** посилання «Вихід» (надпис вже є у списку посилань) на сторінку **logout.php**.
10. Перевірте коректність роботи сценаріїв.

Вправа 6: Створення адміністративних сторінок для керування користувачами

Необхідно створити дві адміністративні сторінки:

6. сторінку списку користувачів (**users.php**),

7. сторінку для додавання нового користувача (**adduser.php**).

- Для того, щоб відобразити список користувачів на сторінці **users.php**, можна створити таблицю, що складається з двох стовпців (ім'я користувача, права на доступ). Пароль не відображається, так як він є конфіденційною інформацією (Прив'язки – Набір записів). Не забудьте організувати повторювану область (Поведінка сервера – Повторити область). Додайте ще декілька користувачів. Перевірте роботу сценарію.

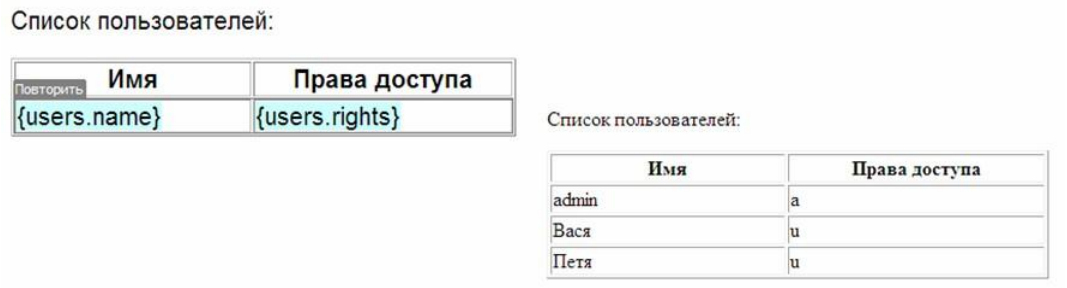

Рис. 10.19. Організація списку користувачів.

- Також додайте посилання на сторінку **adduser.php** для додавання нового користувача (сторінку створимо на наступному кроці) та посилання для повернення на головну сторінку сайту.

- Для додавання користувача на сторінці **adduser.php** створіть форму з необхідними полями для вводу імені, логіна, прав на доступ (a/u) та модель серверної поведінки **Додати запис**.

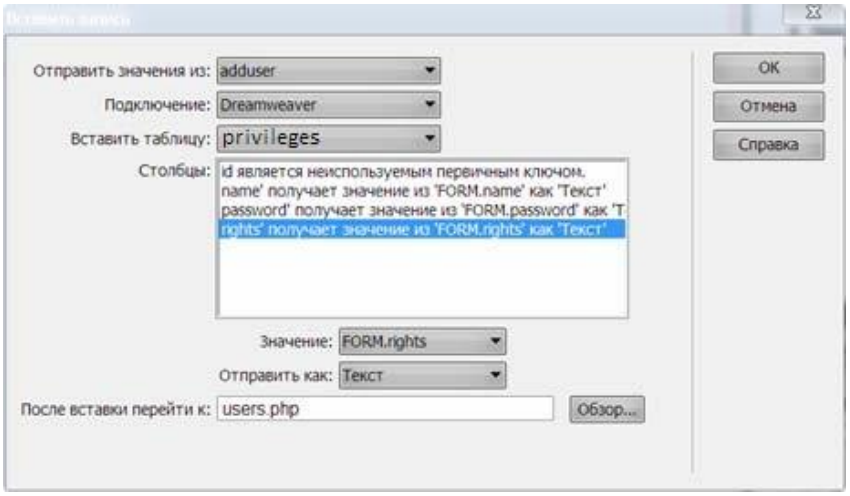

Рис. 10.20. Вікно роботи з записами

- Для перевірки роботи сценарію додайте нового користувача (одного або декількох) з правами користувача (u).

Вправа 7: Адміністративна частина сайту

1. На сторінці **main.php** створіть гіперпосилання на сторінку **users.php** на надписі «Адміністратору».
2. Далі створіть .html сторінку (наприклад, rights.html) з текстом: «Вибачте, Ви не маєте права на перегляд цієї сторінки» та з гіперпосиланням на головну сторінку сайту.
3. Організуйте доступ на сторінку **users.php** тільки для тих, хто має права

адміністратора (тобто права на доступ типу «a»):

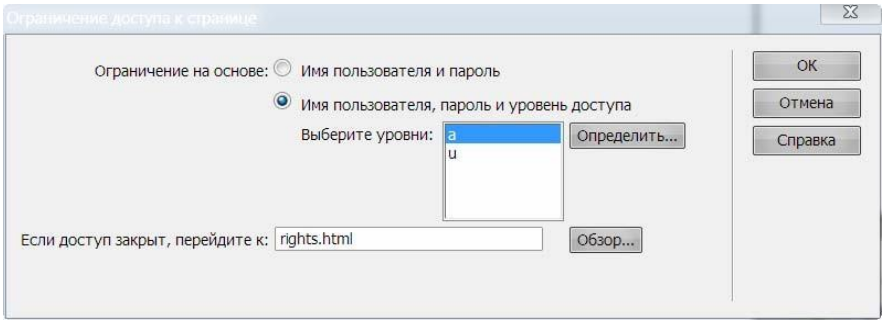

Рис. 10.21. Форма обмеження доступу до сторінки

1. Перевірте коректність виконання сценаріїв. Зайдіть на сайт з облікового запису користувача з користувацькими (не адміністраторськими) правами на доступ та спробуйте перейти за посиланням «Адміністратору». Переконайтеся, що користувач не має доступу для входу на сторінку. Зайдіть на сайт з правами адміністратора та переконайтеся, що в такому випадку ви маєте право на доступ до даної сторінки.

Література

1 Вэллинг Л., Томсон Л. Разработка веб-приложений с помощью PHP и MySQL. М.: Вильямс, 2010. – 848 с.

2 Дамашке Г. PHP и MySQL. М.: НТ-Пресс, 2012 – 320 с.

3 Дронов В. PHP, MySQL и Dreamweaver. Разработка интерактивных Web-сайтов. СПб.: БХВ – Петербург, 2007. – 478 с.

4 Зандстра М. PHP. Объекты, шаблоны и методики программирования. –М.: Вильямс, 2011. – 560 с.

5 Зервас К. Web 2.0. Создание приложений на PHP. М.: Вильямс, 2009. – 544 с.

6 Колисниченко Д.Н. PHP 5/6 и MySQL 6. Разработка Web-приложений.СПб.: БХВ-Петербург, 2010. – 560 с.

7 Кузнецов М., Симдянов И. PHP 5/6. СПб.: БХВ-Петербург, 2009. – 1024 с.

8 Никсон Р. Создаем динамические веб-сайты с помощью PHP, MySQL и JavaScript. Спб.: ПИТЕР, 2011. – 496 с.

9 Пауэрс Д. Adobe Dreamweaver, CSS, Ajax и PHP. .: БХВ-Петербург, 2009. – 1056 с.

10. Пауэрс Д. PHP. Создание динамических страниц. М.: Рид Групп, 2012 – 640 с.

ДОДАТКИ

Додаток 1 Схема сайту «MyTravelNotes»

У процесі послідовного виконання лабораторних робіт створюються декілька статичних сторінок, які є основою сайту, потім розробляються динамічні сторінки для обміну даними з базою даних MySQL, далі статичні сторінки перетворюються в динамічні, додаються динамічні сторінки для роботи з замітками, коментарями, роботою з поштою, обміну файлами з сервером і т.д. Кінцевим етапом роботи є створення сторінок та скриптів для організації розмежування доступу користувачів до сайту з метою забезпечення безпеки.

Додаток 2 Схема бази даних «MySiteDB»

1. Таблиця «notes» містить інформацію про замітки на сайті;
2. Таблица «comments» містить інформацію про коментарі до заміток;
3. Таблица «privileges» містить інформацію про користувачів та їх права на доступ до інформації на сайті.

Додаток 3 Структура таблиць бази даних

***Таблиця 1**. Структура таблиці notes*

Ім'я поля	Опис поля	Тип даних	Інше
id	Ідентифікатор запису	SMALLINT	Ключове поле (AUTO_INCREMENT), INDEX – PRIMARY.
created	Дата створення замітки	DATE	
title	Заголовок замітки	VARCHAR (50)	Рядок фіксованої довжини в 50 символів
article	Вміст замітки	VARCHAR (255)	Рядок фіксованої довжини в 255 символів

Основні атрибути полів:

PRIMARY – визначає поле, що є ключовим атрибутом.
UNSIGNED – забороняє числовим полям приймати від'ємне значення.
AUTO_INCREMENT (A_I) – поле лічильника (окремий тип даних «лічильник» відсутній).
BLOG – дозволяє задати значення поля за замовчуванням.
NOT NULL – визначає обов'язкове для заповнення поле.

***Таблиця 2**. Структура таблиці comments*

Ім'я поля	Опис поля	Тип даних	Інше
id	Ідентифікатор запису	SMALLINT	Ключове поле (AUTO_INCREMENT), INDEX – PRIMARY
created	Дата публікації замітки	DATE	
author	Автор коментаря	VARCHAR (20)	
comment	Вміст коментаря	VARCHAR (255)	Рядок фіксованої довжини в 255 символів
art_id	Зовнішній ключ для прив'язки коментаря до замітки в таблиці Notes	SMALLINT	

Таблиця 3. *Структура таблиці privileges*

Ім’я поля	Опис поля	Тип даних
id	Id запису	SMALLINT, ключевое поле, AUTO_INCREMENT
name	VARCHAR (20)	Ім’я користувача
password	VARCHAR (20)	Пароль користувача
rights	VARCHAR(1)	Права на доступ користувача до сторінок сайту

Останнє поле rights може вміщувати тільки один символ, що показує права користувача на доступ:

a – права на доступ адміністратора;
u – права на доступ користувача.

Додаток 4 Основні відомості про роботу з базою даних

База даних – структуроване сховище деякого набору даних деякої предметної області в іменованій частині жорсткого диску.

СУБД (Система Управління Базою даних) – це програмне забезпечення, що має засоби обробки на мові БД та керує доступом до БД. Воно зберігає дані та надає користувачам можливість отримувати та модифікувати ці дані.

Реляційні БД зберігають інформацію у вигляді зв'язаних один з одним таблиць. Таблиця складається з полів (стовпців), записів (рядків) та комірок.

Обов'язковим елементом таблиці є ключ. Ключ – це поле, що однозначно визначає значення всіх інших полів у таблиці. Ключ завжди унікальний, тобто в будь-який момент часу таблиця БД не може вміщувати жодні два записи, що мають однакові значення ключових полів.

В реляційній базі даних таблиці пов'язані один з одним. Зв'язки бувають трьох типів:

- Один-до-одного (1:1);
- Один-до-багатьох (1:M);
- Багато-до-багатьох (N:M).

Основні поняття мови SQL

SQL — (Structured Query Language) це мова структурованих запитів. Мова SQL була спеціально розроблена для взаємодії з базами даних.

Основні оператори визначення ***структури даних***:

- CREATE (створення);
- ALTER (зміна);
- DROP (видалення).

Основні оператори роботи з ***даними***:

- SELECT (вибір);
- INSERT (вставка);
- UPDATE (оновлення);
- DELETE (видалення).

Основні функції PHP для роботи з MySQL

При роботі з БД ***в першу чергу*** *встановлюється з'єднання з сервером.*

Функція з'єднання з сервером (*параметри функції: ім'я сервера, ім'я користувача, пароль*):

```
//Присвоювання значень змінним, що необхідні для підключення до сервера:
ім'я сервера, ім'я користувача, пароль.
$hostname = 'localhost';
$user = 'administrator';
$password = 'admin_password';

//Функція встановлення з'єднання з сервером  mysqli_connect(). Параметри
```

функції: ім'я сервера, ім'я користувача, пароль.
$link = ***mysqli_connect*** *($hostname, $user, $password);*

Варіант з літеральними строками замість змінних:
$link = mysqli_connect ('localhost', 'administrator', 'admin_password');

Важливо: пароль є **обов'язковим** параметром функції mysqli_connect(). Якщо пароля немає, йому присвоюється пусте значення:

$hostname = 'localhost';
$user = 'administrator';
$password = ' ';
$link = mysqli_connect ($hostname, $user, $password);

Варіант з літеральними строками замість змінних:
$link = mysqli_connect ('localhost', 'administrator', ' ');

Вибір бази даних

Після встановлення з'єднання з сервером потрібно вибрати БД для подальшої роботи.

Функція вибору БД для подальшої роботи (*параметри функції: ім'я з'єднання з сервером, ім'я БД):*

mysqli_select_db*($link, $database);*

Варіант з літеральними строками замість змінної:
mysqli_select_db($link, 'MyDatabase');

Виконання запитів MySQL

Функція PHP **mysqli_query()** дозволяє використовувати основні оператори мови SQL (SELECT, INSERT, UPDATE, DELETE) для звернення до БД.

Параметрами функції mysqli_query() є з'єднання з сервером та SQL-запит.

Наприклад:

//з'єднання з сервером
$link = mysqli_connect ($hostname, $user, $password);
//формування SQL-запиту
$query = "CREATE DATABASE MyDatabase";
//Виконання функції mysqli_query() для реалізації запиту
$result =* ***mysqli_query****($link, $query);

Або

$link = mysqli_connect ($hostname, $user, $password);
$result =* ***mysqli_query****($link, "CREATE DATABASE MyDatabase");
//При такому використанні функції сам запит передається прямо в функцію, без «проміжної» змінної $query (як у попередньому прикладі). Перший варіант є кращим, тому що полегшує читабельність коду (SQL-запити можуть бути досить довгими).