Веб-технології

Лабораторний практикум

Міністерство освіти і науки України
Вінницький національний технічний університет

Веб-технології

Лабораторний практикум

Вінниця
ВНТУ
2016 рік

УДК 004.774(075)
ББК 32.973.22я73
В26

Надано гриф «Лабораторний практикум. Електронне видання» та рекомендовано до використання в навчальному процесі Вченою радою Вінницького національного технічного університету Міністерства освіти і науки України (протокол № 5 від 24 листопада 2016 р.).

Рецензенти:

А. Я. Кулик, доктор технічних наук, професор
О. М. Василевський, доктор технічних наук, професор
В. В. Ковтун, кандидат технічних наук, доцент

В26 **Веб-технології** : лабораторний практикум / Бісікало О. В., Маслій Р. В., Паламарчук Є. А., Яцковська Р. О. – Вінниця : ВНТУ, 2016. – 116 с.

Лабораторний практикум присвячений популярній відкритій технології LAMP. Розглядаються основні поняття веб-технологій, процедури інсталяції веб-сервера Apache та сервера бази даних MySQL, особливості мов програмування сценарного типу PHP та Python. Крім цього у практикумі представлені альтернативні методи автоматизованого створення веб-ресурсів.

Лабораторий практикум призначений для студентів спеціальностей 122 – комп'ютерні науки та інформаційні технології, 151 – автоматизація та комп'ютерно-інтегрованих технологій та 152 – метрологія та інформаційно-вимірювальна техніка.

УДК 004.774(075)
ББК 32.973.22я73

ЗМІСТ

ЛАБОРАТОРНА РОБОТА №1. СТВОРЕННЯ ЛАБОРАТОРНОГО СЕРЕДОВИЩА

Мета: ознайомлення з принципами побудови експериментального середовища для вивчення WEB–технологій, одержання навичок інсталяції Ubuntu – популярного дистрибутива Linux та пакету Denwer під операційну систему Windows, дослідження основних можливостей модуля phpMyAdmin для адміністрування бази даних MySQL.

Короткі теоретичні відомості

Linux (повна назва GNU/Linux) – загальна назва UNIX-подібних операційних систем на основі однойменного ядра та зібраних для нього бібліотек і системних програм, розроблених у рамках проекту GNU.

До операційної системи GNU/Linux також часто відносять програми, що доповнюють цю операційну систему, а також прикладні програми, які фактично перетворюють її на повноцінне багатофункціональне операційне середовище.

На відміну від більшості інших операційних систем, GNU/Linux не має єдиної «офіційної» комплектації. Замість цього GNU/Linux постачається у великій кількості так званих дистрибутивів, в яких програми GNU з'єднуються з ядром Linux та іншими програмами. Найбільш відомими дистрибутивами GNU/Linux вважаються Slackware, Debian GNU/Linux, Red Hat, Fedora, Mandriva, SuSE, Gentoo, Ubuntu.

Більшість користувачів для встановлення GNU/Linux застосовують саме дистрибутиви. Це викликано тим, що дистрибутив не є простим набором програм, а є рядом рішень для різних задач користувачів, об'єднаних єдиними системами установки, управління та обновлення пакетів, налагодження та підтримки.

Ubuntu – операційна система для робочих станцій, лептопів і серверів, є одним з найпопулярніших у світі дистрибутивом Linux. Серед основних цілей Ubuntu – надання сучасного і водночас стабільного програмного забезпечення для пересічного користувача з помітним акцентом на простоту встановлення і користування. Серверний варіант системи включає також засоби, потрібні для організації сервера баз даних, веб-сервера, сервера електронної пошти тощо.

Зручним інструментом для відлагодження WEB-додатків зарекомендував себе пакет "Джентльменський набір web-розробника" ("Д.н.w.р", читається як "Денвер") – набір дистрибутивів (Apache, PHP, MySQL, Perl тощо) й програмна оболонка. Денвер використовується web-розробниками для створення сайтів на "домашній" (локальній) windows-машині без необхідності виходу в Інтернет. Головна особливість Денвера – зручність при від-

даленій роботі відразу над декількома незалежними проектами та можливість розміщення на Flash-накопичувачі.

До складу базового пакета Денвер входять:

- Apache 2 з підтримкою SSL і mod_rewrite;
- PHP 5: файли для виконання, модуль для веб-сервера Apache, дистрибутивний і адаптований конфігураційний файл, бібліотека GD, модулі підтримки MySQL і sqLite;
- MySQL 5 з підтримкою InnoDB, трансакцій і кириличних кодувань (windows-1251);
- phpMyAdmin - панель керування базою даних MySQL, а також скрипт (сценарій), який спрощує додавання нового користувача MySQL;
- загальнокласове налагодження sendmail (/usr/sbin/sendmail), яке не відправляє листа, а записує його у директорію /tmp/!sendmail;
- система автоматичного пошуку віртуальних хостів і відновлення системного файлу hosts, а також конфігурації Apache. Завдяки їй додавання нового віртуального хоста (або домена третього рівня) полягає в простому створенні каталогу в /home (див. за аналогією з уже існуючими хостами) і перезапуск комплексу. Всі зміни вносяться в конфігураційні та системні файли автоматично, але ви можете керувати цим процесом за допомогою механізму шаблонів хостів (див. /usr/local/apache/conf/httpd.conf за детальними роз'ясненнями).

Домашня сторінка інструментарію Денвер знаходиться за адресою http://www.denwer.ru/, де і можна завантажити на власний комп'ютер останню версію базового пакету та пакети розширень (http://rus.denwer.net/), отримати необхідну довідкову інформацію. Після інсталяції дистрибутиву створюється віртуальний диск (як правило Z:\), зміст якого повністю відповідає папці D:\WebSerwers.

Апробація базової WEB-технології PHP-MySQL здійснюється в оболонці Денвер шляхом запуску файлів з PHP-сценаріями (*.PHP) на WEB-сервері Apache в умовах підтримки баз даних діючим MySQL-сервером. При цьому робоча область WEB-серверу Apache – Z:\home\localhost\www\ або D:\WebSerwers\home\localhost\www\, а робоча папка для баз даних MySQL – Z:\usr\local\mysql5\data\ або D:\WebSerwers\usr\local\mysql5\data\. Перегляд потрібних файлів *.PHP здійснюється у будь-якому встановленому на комп'ютері браузері за адресою http://localhost/ або http://127.0.0.1/.

Хід роботи

1. Встановити дистрибутив Ubuntu операційної системи Linux. З цією метою завантажити останню версію дистрибутива за адресою http://ubuntu.net/. Далі послідовно виконати всі наступні кроки програми-майстра згідно рисунків 1.1-1.6 та встановити Ubuntu.

2. Зайти на сайт Денвер за адресою http://www.denwer.ru/ та, за необхідністю, передивитися відео уроки – Денвер: які бувають WEB-сервери та що це таке (http://vimeo.com/8952074), Денвер: що таке локальний сервер (http://vimeo.com/8952237), Денвер: встановлення (http://vimeo.com/8952406) та Денвер: перший запуск та приклади використання (http://vimeo.com/8952621).

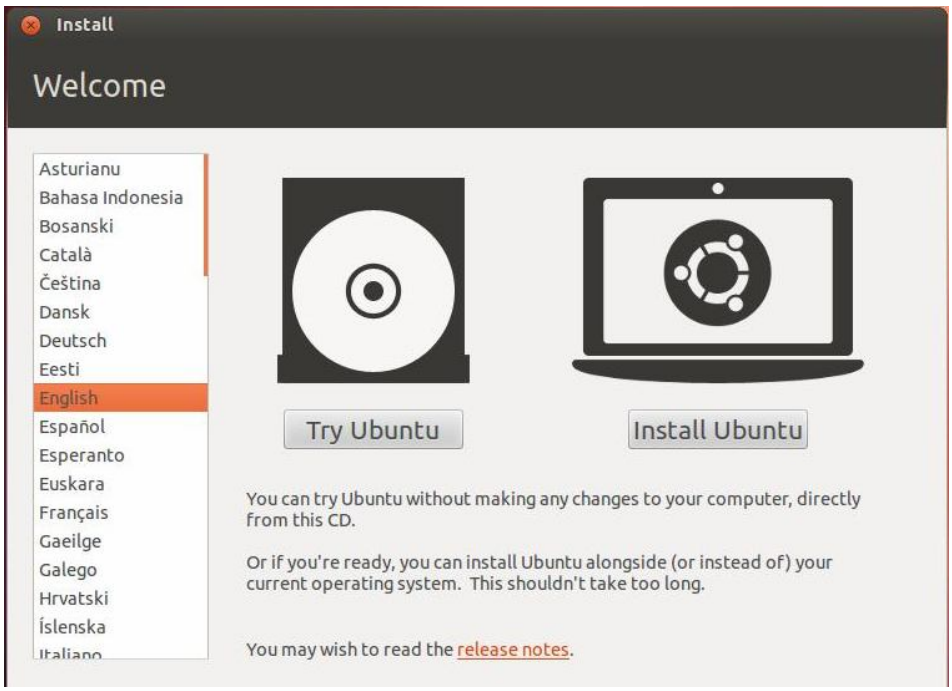

Рисунок 1.1 – Вікно вибору мови для Ubuntu

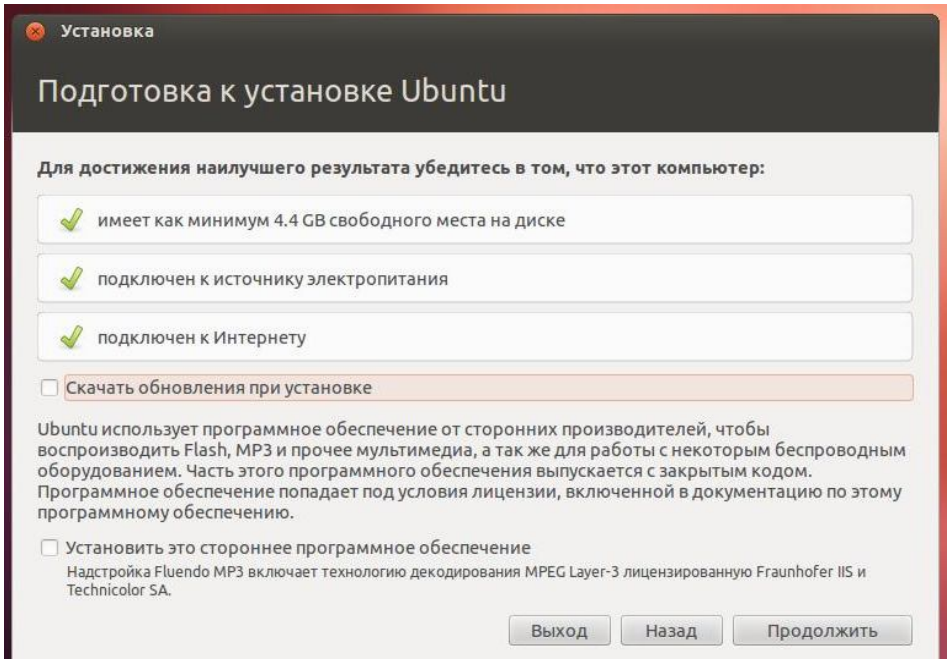

Рисунок 1.2 – Вікно підготовки до встановлення Ubuntu

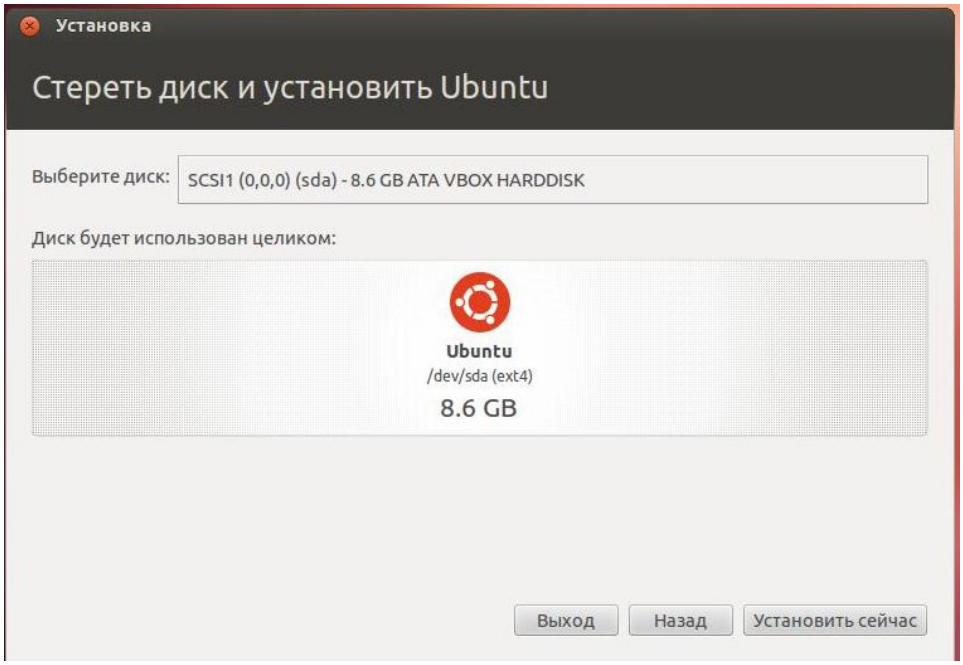

Рисунок 1.3 – Вікно встановлення Ubuntu на окремий жорсткий диск

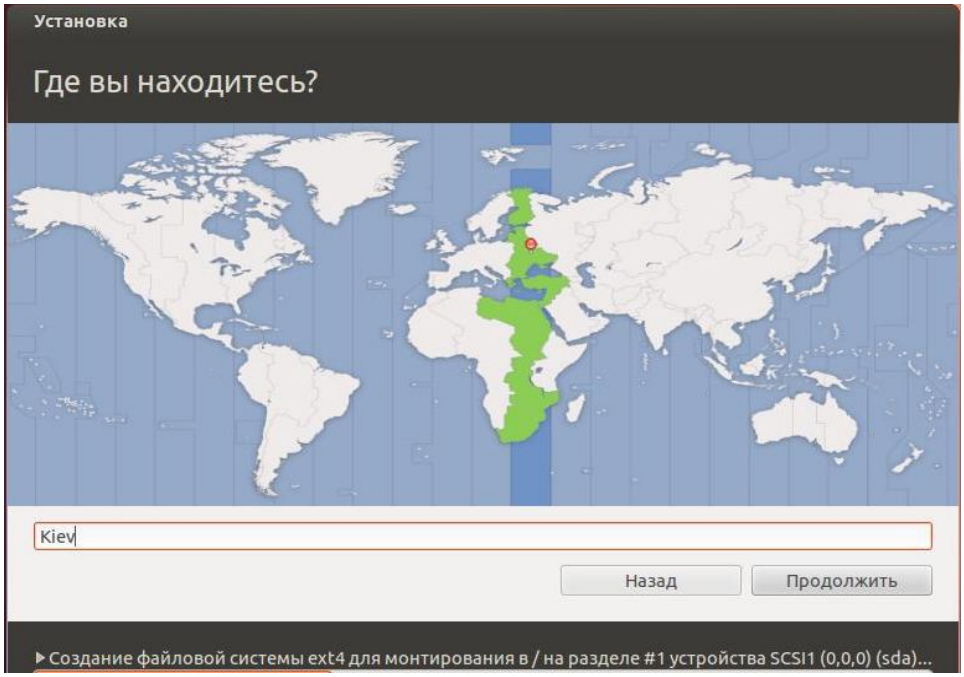

Рисунок 1.4 – Вікно вибору часового поясу для Ubuntu

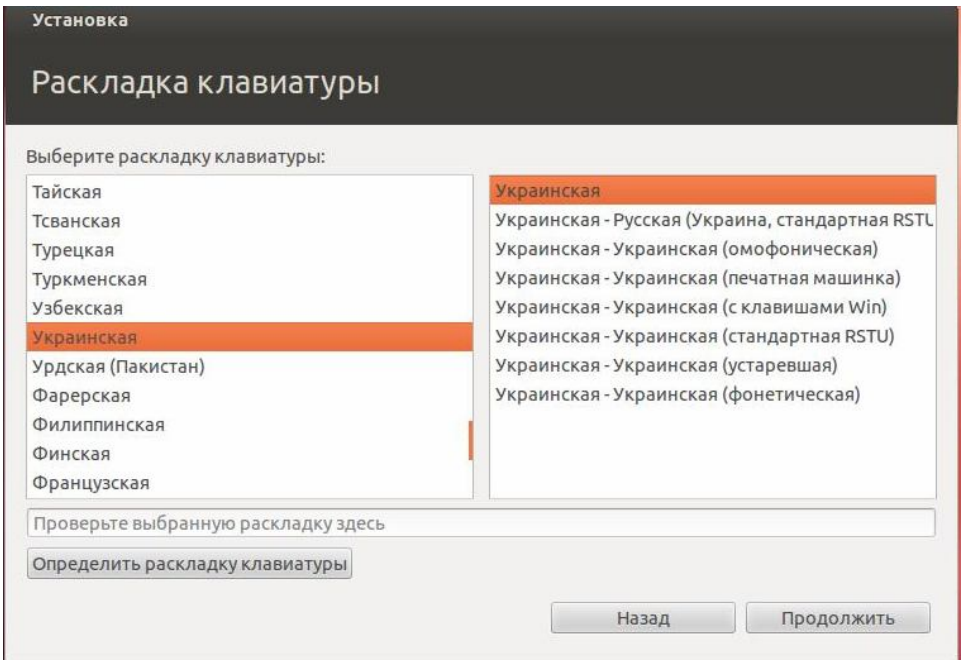

Рисунок 1.5 – Вікно вибору розкладки клавіатури для Ubuntu

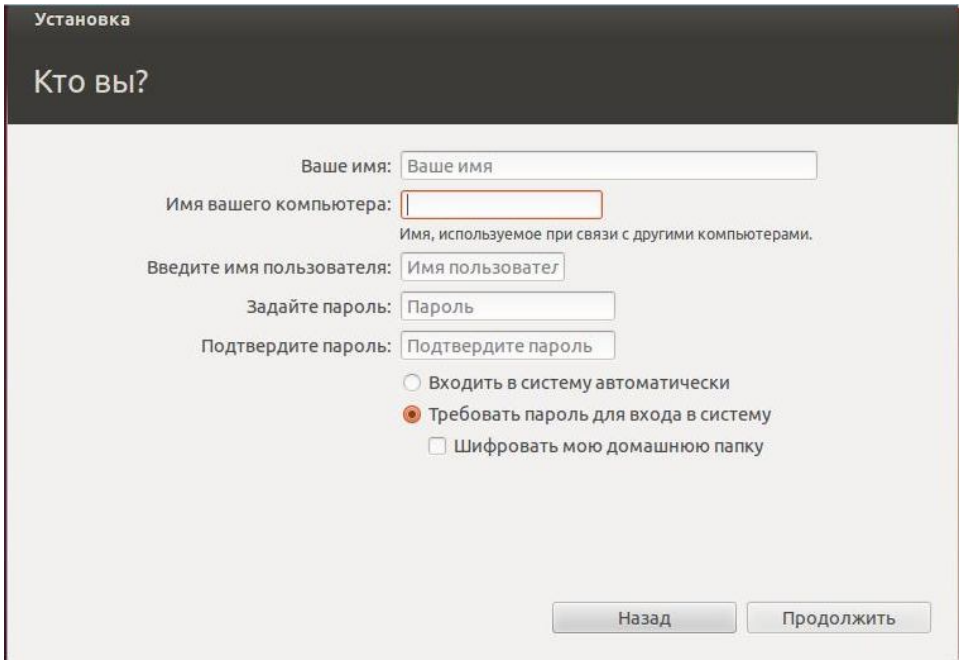

Рисунок 1.6 – Вікно введення персональних даних для Ubuntu

3. Завантажити останню версію дистрибутива Денвер за адресою http://rus.denwer.net/. Інсталювати дистрибутив на власному комп'ютері під операційною системою Windows.

4. Запустити оболонку Денвер за допомогою ярлика Start Denwer . На екрані має з'явитися тимчасове вікно ініціалізації пакету (рисунок 1.7).

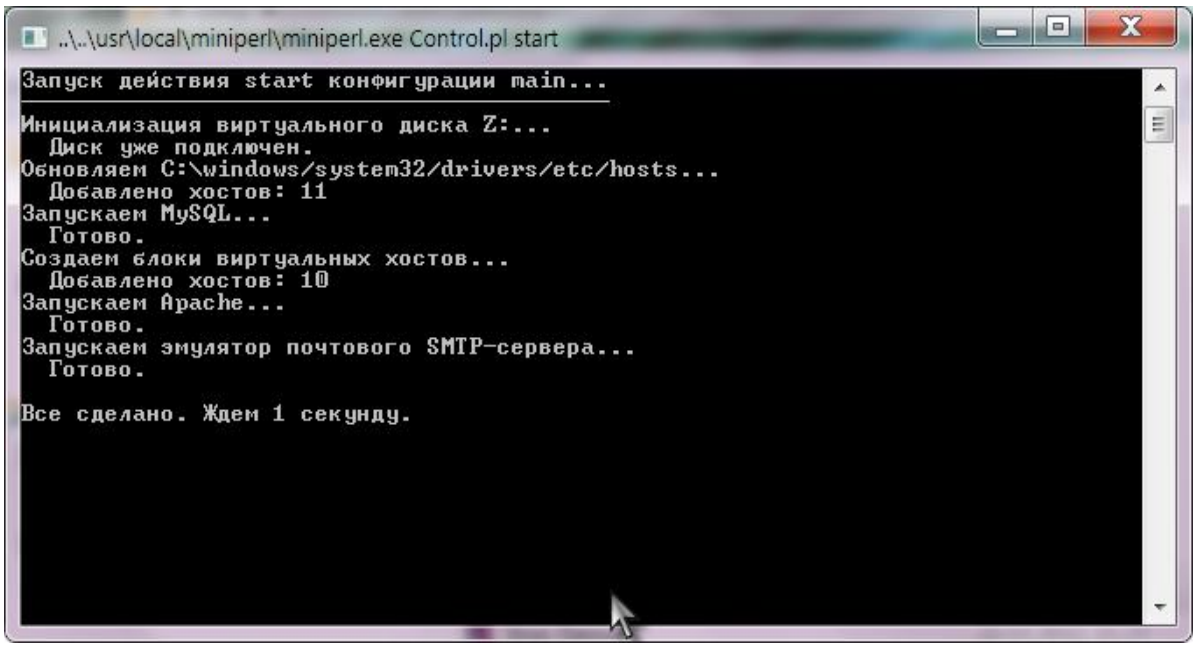

Рисунок 1.7 – Вікно ініціалізації пакету Денвер

За умови безпомилкового виконання попередніх дій введення у браузері адреси http://localhost/ призведе до появи стартового вікна оболонки (рисунок 1.8), що свідчитиме про дієздатність WEB-серверу Apache.

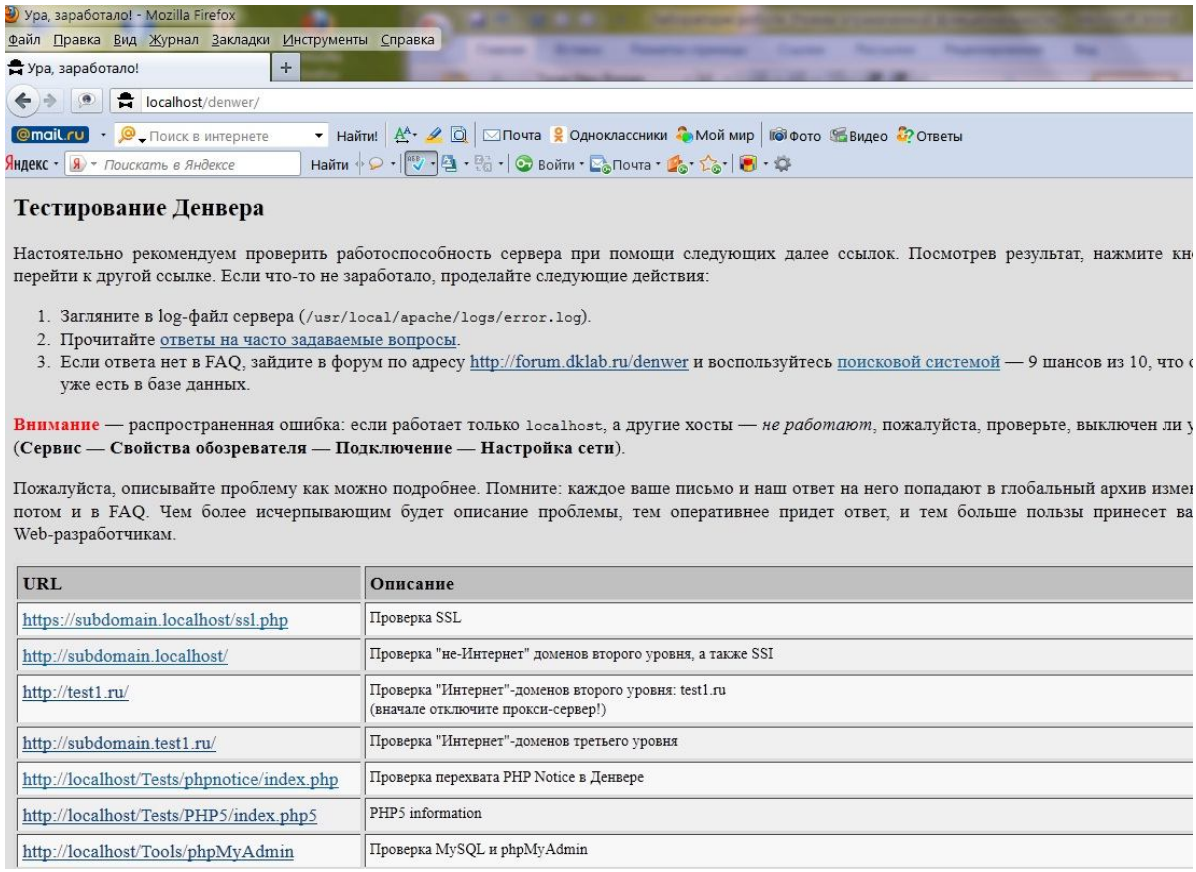

Рисунок 1.8 – Стартове вікно пакету Денвер

5. Перевірити дієздатність MySQL-серверу та пакету phpMyAdmin. Для цього перейти у вікні браузера за адресою http://localhost/Tools/phpMyAdmin/ та пересвідчитися (рисунок 1.9) у доступності 17-ти таблиць бази даних information_schema. Безпосередньо у браузері провести дослідження основних функціональних можливостей пакету phpMyAdmin.

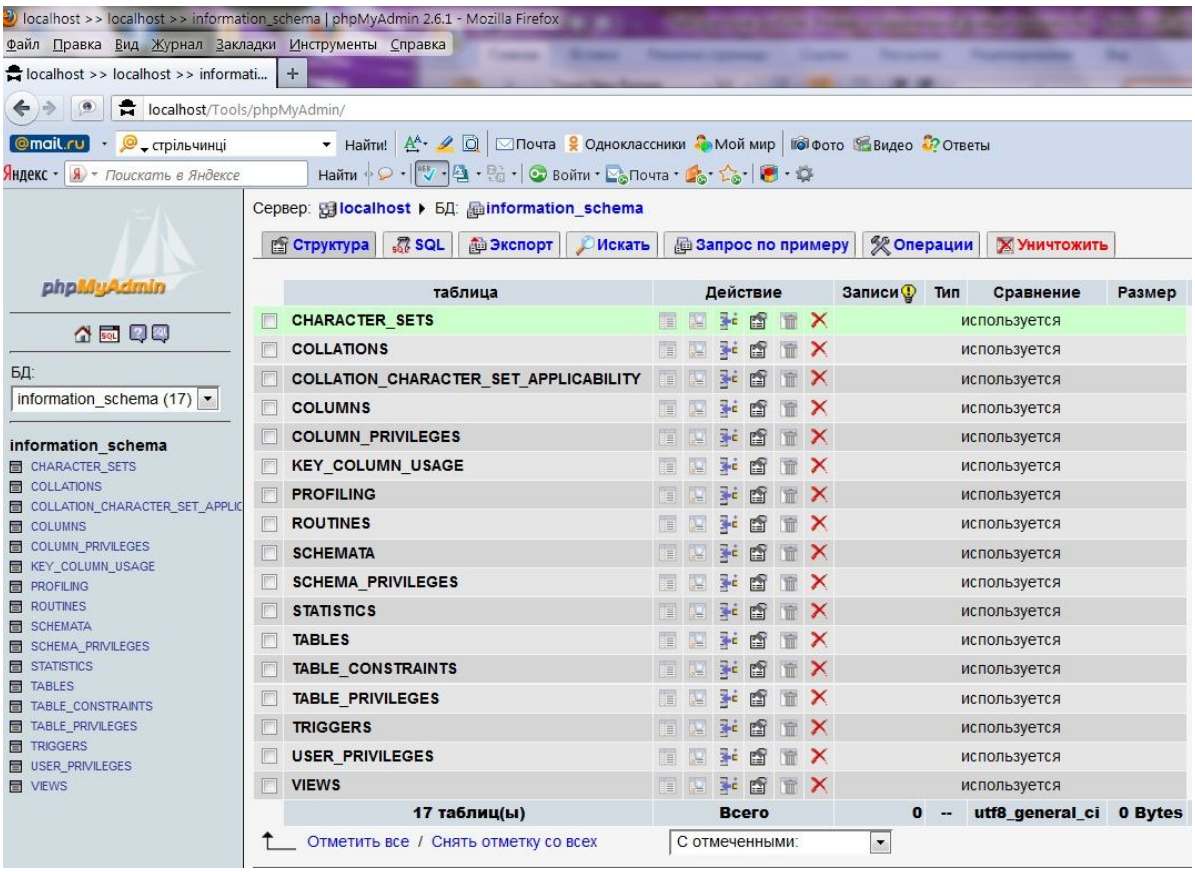

Рисунок 1.9 – База даних information_schema у пакеті phpMyAdmin

6. Остаточну апробацію технології PHP-MySQL у цілому здійснити таким чином:

а) створити у папці Z:\home\localhost\www\ файл create_tables.php з таким змістом (рисунок 1.10);

б) створити у папці Z:\home\localhost\www\ файл inc.php з таким змістом (рисунок 1.11);

в) запустити у браузері сценарій створення 2-х таблиць (Pictures, Words) у базі даних automation шляхом переходу за адресою http://localhost/create_tables.php. Перевірити результати виконання сценарію за допомогою пакету (http://localhost/Tools/phpMyAdmin/) та перегляду вмісту папки Z:\usr\local\mysql5\data\ .

```
<?php
include('inc.php');  // приєднуємо файл з інформацією про підключення
include('inc_loc.php');
$db = mysql_connect($cServname,$cUsername,$cPassword) // під'єдна    тись до MySql
сервера
or die('Unable to connect to MySQL server.'); // або вивести по-відомлення про немож-
ливість під'єднання
if (!mysql_select_db( $cDatabase, $db )) // якщо неможливо вибрати базу даних (тобто
її не існує), то
{
 mysql_query( "CREATE DATABASE `$cDatabase`", $db ); // ми її створюємо.
 }
 mysql_select_db( $cDatabase, $db ) or die ("Could not activate the database");
 // створюємо таблицю Pictures
 mysql_query("CREATE TABLE `$cTablename1`
 (id int(11) auto_increment,
 picture varchar(100) default NULL,
 category varchar(100) default NULL,
 URL varchar(100) default NULL,
 PRIMARY KEY(id))",$db) or die(mysql_error()); // якщо запит не виконано – виво-
 димо зміст помилки
 echo"<h4>Таблиця $cTablename1 успішно створена</h4>";
 // створюємо таблицю Words
 mysql_query("CREATE TABLE `$cTablename2`
 (id int(11) auto_increment,
 word varchar(100) default NULL,
 IDp int(11) default NULL,
 PRIMARY KEY(id))",$db) or die(mysql_error());// якщо запит не виконано – виводи-
 мо зміст помилки
 echo"<h4>Таблиця $cTablename2 успішно створена</h4>";
 mysql_close($db);
 ?>
```

Рисунок 1.10 – Вміст файлу create_tables.php

```
<?php
$cServname = "localhost";
$cUsername = "root";
$cPassword = "";
$cDatabase = "automation";
?>
```

Рисунок 1.11 – Вміст файлу inc.php

7. Запустити операційну систему Ubuntu та самостійно вивчити довідкову документацію щодо пакету phpMyAdmin. Після цього виконати пп.5-6 в операційній системі Ubuntu.

Вміст звіту

а) тема і мета лабораторної роботи;
б) у відповідності до кроків ходу роботи навести:
   1) графічне представлення інженерних дій у вигляді схем програм, схем взаємодії програмних модулів, діаграм використання або інших UML-діаграм;
   2) лістинг всіх відлагоджених програмних модулів;
   3) результати виконання програмних модулів;
в) змістовні висновки за результатами роботи.

Питання для самоконтролю

1. Дайте загальну характеристику операційної системи Linux.
2. Що таке дистрибутив Linux?
3. Перерахуйте та охарактеризуйте найбільш відомі дистрибутиви Linux.
4. Що потрібно визначити для встановлення дистрибутива Ubuntu операційної системи Linux?
5. Що входить до складу пакету Денвер?
6. Яким чином інсталювати пакет Денвер?
7. Як перевірити працездатність WEB-серверу Apache?
8. Як перевірити працездатність MySQL-серверу?
9. Які основні функції підтримує пакет phpMyAdmin?
10. Як перевірити дієздатність технології PHP-MySQL у цілому?

ЛАБОРАТОРНА РОБОТАЛ №2. ХОСТИНГ ТА FTP-ДОСТУП

Мета: ознайомлення з основними поняттями хостингу, одержання навичок у FTP-доступі до WEB-ресурсів на сервері, дослідження можливостей основних інструментальних засобів FTP-доступу в операційних системах Windows і Linux.

Короткі теоретичні відомості

Хо́стинг (англ. *hosting*) – послуга, що надає дисковий простір для розміщення фізичної інформації на сервері. Зазвичай під поняттям послуги хостингу мають на увазі, як мінімум, послугу розміщення файлів сайту на сервері, на якому запущене програмне забезпечення, необхідне для обробки запитів до цих файлів (WEB-сервер). Як правило, до послуг хостингу вже входить надання місця для поштової кореспонденції, баз даних, DNS файлового сховища тощо, а також підтримка функціонування відповідних сервісів, однак вони можуть надаватися і окремо.

За умовами надання хостинг часто розділяється на платний і безкоштовний. Зазвичай компанія, що надає безкоштовний хостинг, заробляє шляхом показу реклами на сторінках, розміщених на ньому. Безкоштовний хостинг, як правило, повільніше платного, надає тільки базові послуги і ненадійний. Приватні особи для своїх домашніх сторінок на початковому етапі їх розвитку використовують безкоштовний хостинг. Громадські організації можуть використовувати як платний хостинг, так і безкоштовний. Комерційні організації практично завжди користуються послугами платного хостингу.

Види хостингу

Повнофункціональний хостинг.

Віртуальний сервер – надається місце на диску для розміщення WEB-сайтів, середовище виконання WEB сервісів єдине для багатьох користувачів, ресурси розподілені між усіма користувачами на одному сервері, де може розміщуватись від 50 до 1000 користувачів. Невеликі дешеві хостинг-провайдери часто нехтують безпекою і взагалі не розмежовують привілеїв користувачів, що дозволяє одному користувачеві на сервері мати доступ до сайтів сотень інших користувачів.

Віртуальний виділений сервер (VPS або VDS) – послуга, у рамках якої користувачеві надається так званий віртуальний виділений сервер. Доступна для управління операційна система здебільшого відповідає фізично встановленій на виділеному сервері. Зокрема це стосується прав адміністр-

ратора, root-доступу, власних IP-адрес, портів, правил фільтрування і таблиці маршрутизації.

Виділений сервер – користувачеві повністю надається сервер. Використовується для реалізації нестандартних завдань (сервісів), а також розміщення «важких» веб-проектів, які не можуть співіснувати на одному сервері з іншими проектами і вимагають під себе всі ресурси сервера.

Колокація (Collocation) – надання місця в дата-центрі провайдера для обладнання клієнта (зазвичай шляхом монтажу в стійці) та підключення його до Інтернету.

Обмежений або хостинг певних програм (послуг):

- веб хостинг;
- E-mail хостинг;
- DNS хостинг;
- ігровий хостинг;
- Wiki farm.

У Додатку наведено перелік основних безкоштовних хостингів та послуг, які вони надають.

Протокол передачі файлів (англ. *File Transfer Protocol, FTP*) — дає можливість абоненту обмінюватися двійковими і текстовими файлами з будь-яким комп'ютером мережі, що підтримує протокол FTP. Установивши зв'язок з віддаленим комп'ютером, користувач може скопіювати файл з віддаленого комп'ютера на свій, або скопіювати файл з свого комп'ютера на віддалений.

Переваги FTP-клієнта. Навіть якщо в адмінці передбачена завантаження необмеженої кількості файлів одноразово, існує встановлене сервером обмеження обсягу даних, переданих формою (зазвичай до 8 мегабайт) і обмеження на розмір кожного файлу, що завантажується по HTTP-протоколу (зазвичай до 2 мегабайт). Отже, звичайно користувач не може завантажити файли розміром більше 2 мегабайт, а сумарний розмір файлів з урахуванням кодування не повинен перевищувати 8 мегабайт. А на основі використання FTP-клієнта користувач може одним клацанням миші відправити на сервер повністю підготовлений сайт, який може мати і 10 тисяч файлів (наприклад, CMS Joomla), а не тисячократно повторювати завантаження файлів по HTTP-протоколу.

Хід роботи

1. Розглянути роботу по управлінню хостингом на прикладі вміщеного першим у таблиці Додатку сервісу hostinger.com.ua. Провести реєстрацію на сайті та перейти на вкладку «Домены», де створити безкоштовне доменне ім’я. Для управління хостингом перейти на вкладку «Панель» (рисунок 2.1), де ви можити побачити створене нове доменне ім’я.

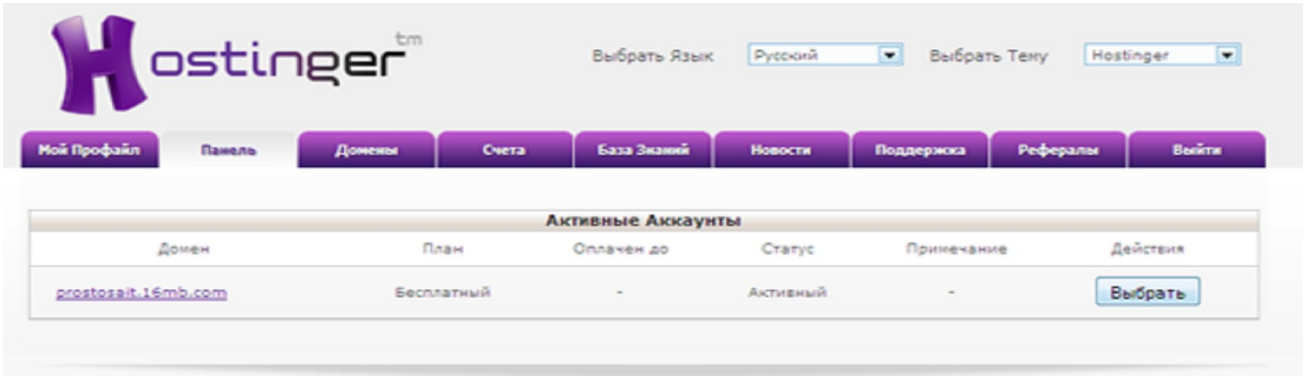

Рисунок 2.1 – Реєстрація доменного імені на hostinger.com.ua

2. Натиснути на кнопку «Выбрать» і переглянути всі можливі служби для роботи з вашим хостингом, а саме служби для роботи з файлами, базами даних, електронною поштою, доменними іменами, онлайн-конструктором сайтів тощо (рисунки 2.2-2.3). Розглянути деякі з найбільш важливих служб:

а) FTP доступ – тут ви можете радагувати параметри доступу по ftp згідно з рисунком 2.4.

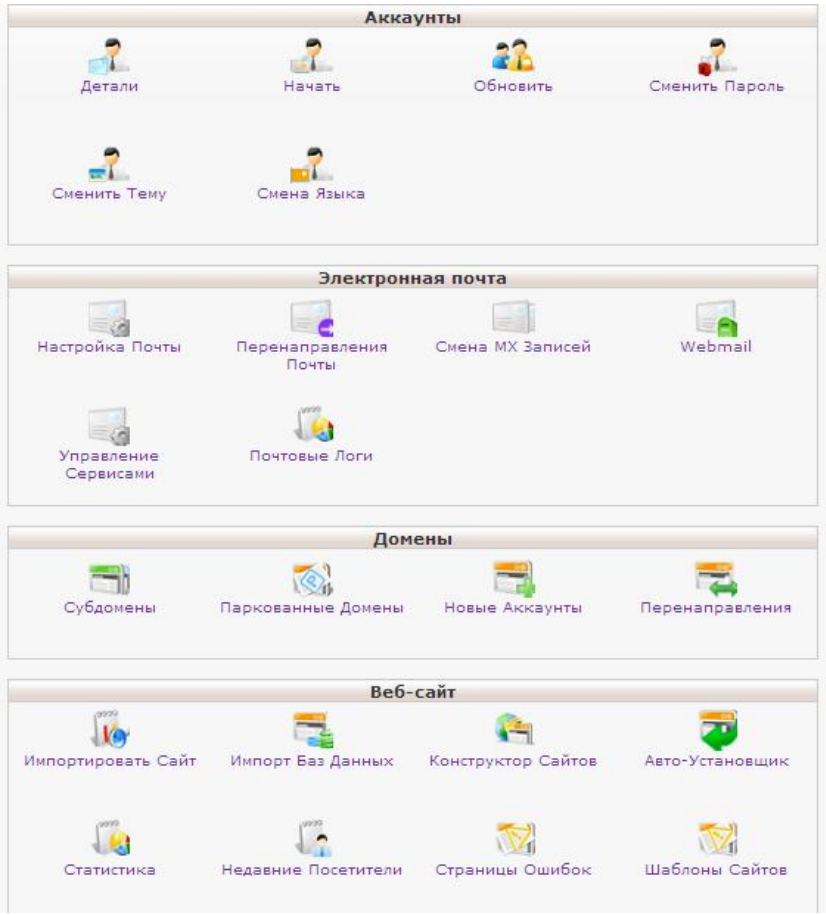

Рисунок 2.2 – Служби «Аккаунты», «Электронная почта», «Домены» та «Веб-сайт»

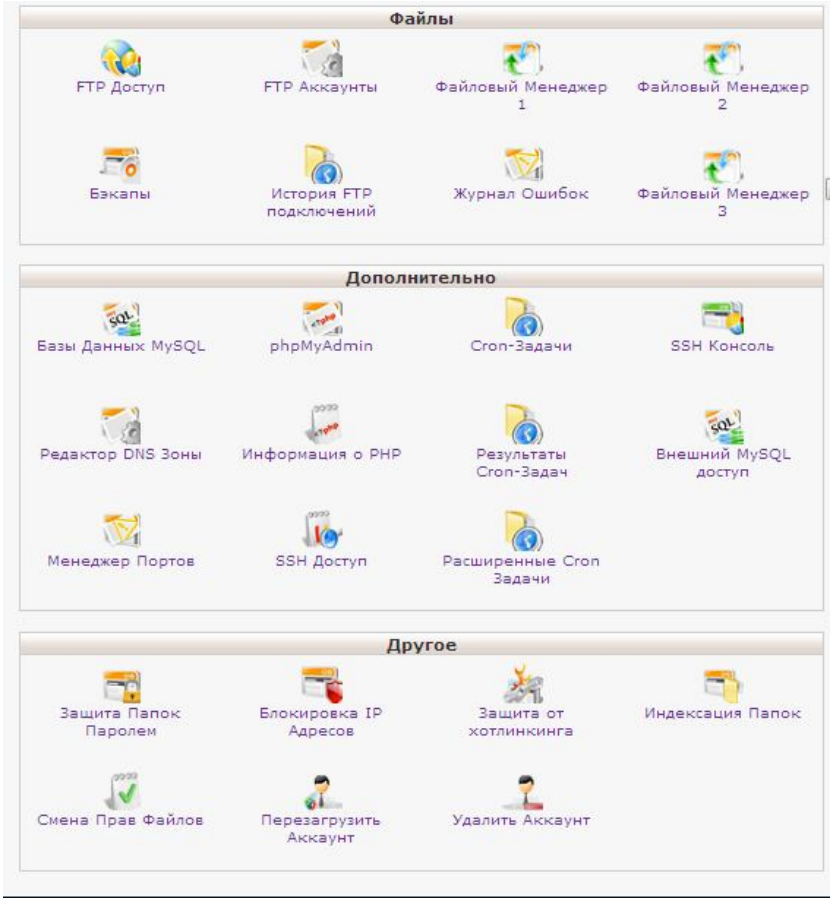

Рисунок 2.3 – Служби «Файлы», «Дополнительно» та «Другое»

FTP Доступ	
FTP сервер	prostosait.16mb.com
FTP IP	31.170.164.130
FTP порт	21
FTP пользователь	u477861564
FTP пароль	••••••••••
Папка для загрузки файлов	public_html
Забыли свой FTP пароль?	Сменть пароль аккаунта
Рекомендуемые FTP клиенты	SmartFTP or FileZilla

Рисунок 2.4 – Редагування параметрів доступу по ftp

Служба для менеджера файлів подібна до Провідника Windows (рисунок 2.5).

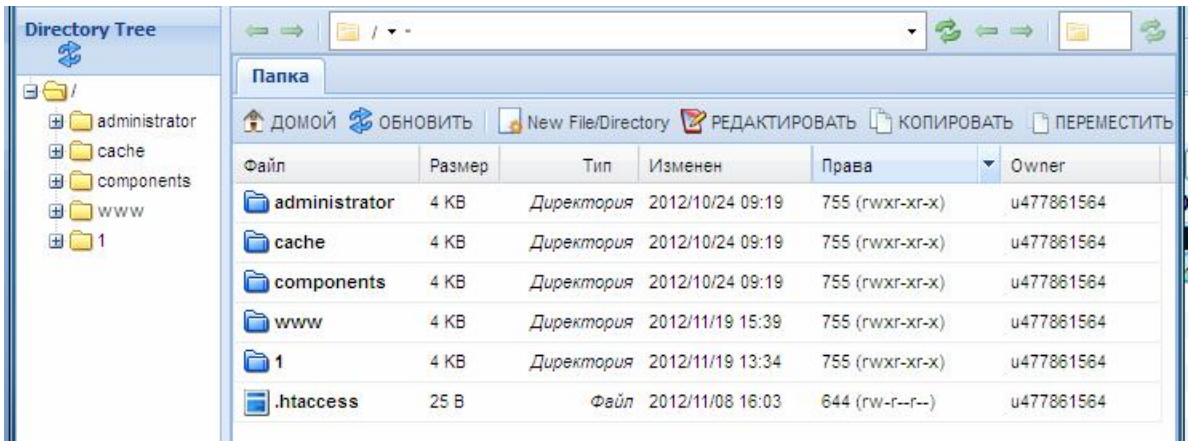

Рисунок 2.5 – Служба «Менеджер файлів»

«Базы Данных MySQL» – служба для створення, видалення, редагування та управління доступом до баз даних MySQL (рисунок 2.6).

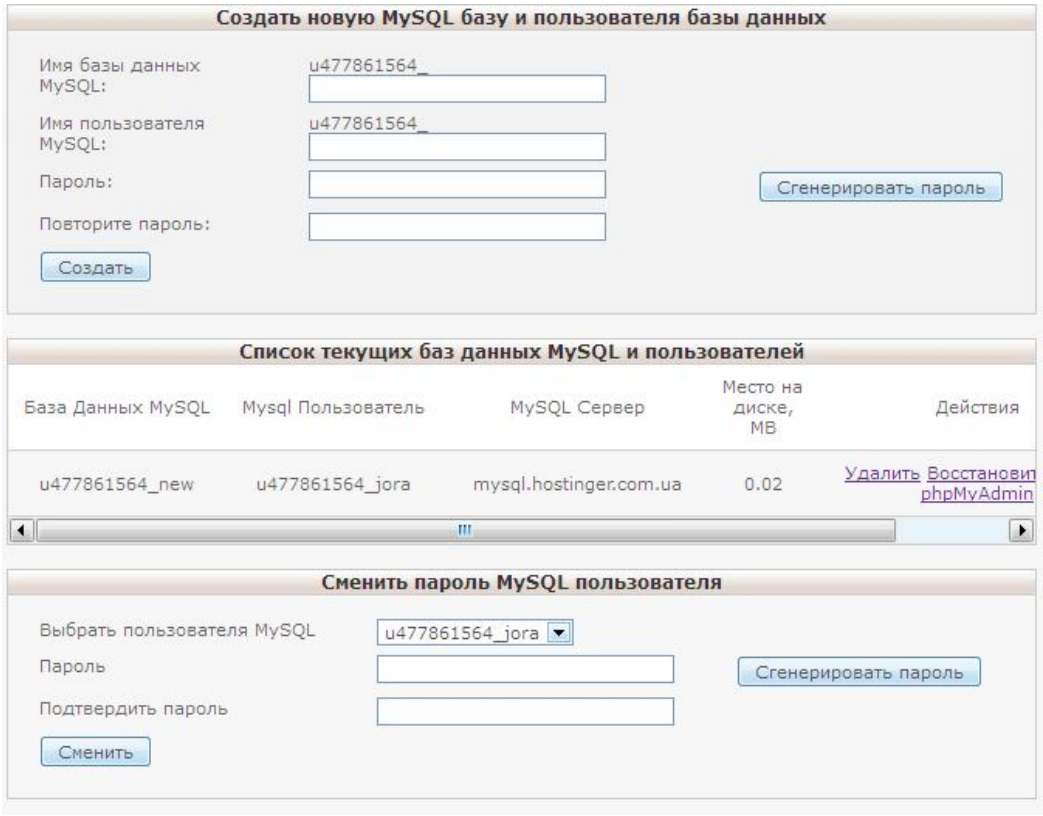

Рисунок 2.6 – Служба «Базы Данных MySQL»

Нарешті, охарактеризований в лабораторній роботі №1 phpMyAdmin – зручна служба для роботи з уже створеними базами даних (рисунок 2.7). Тут ви зможете повноцінно працювати з вибраною базою даних на віддаленому сервері.

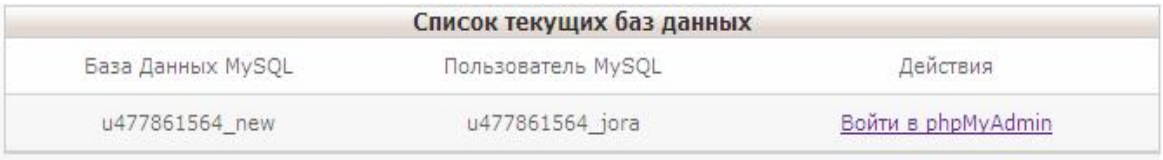

Рисунок 2.7 – Служба «phpMyAdmin»

3. Натиснути кнопку «Войти в phpMyAdmin» та перевірити основні можливості цієї служби згідно пп.5-6 завдання до лабораторної роботи №1.

4. Вивчити чотири альтернативні можливості створення FTP-клієнта за допомогою загальнодоступних інструментальних засобів.

4.1 Total commander**.** Увійти у Total Commander та відкрити у головному меню (рисунок 2.8) «Сеть» – «Соединиться с FTP-сервером» (або натиснути CTRL + F).

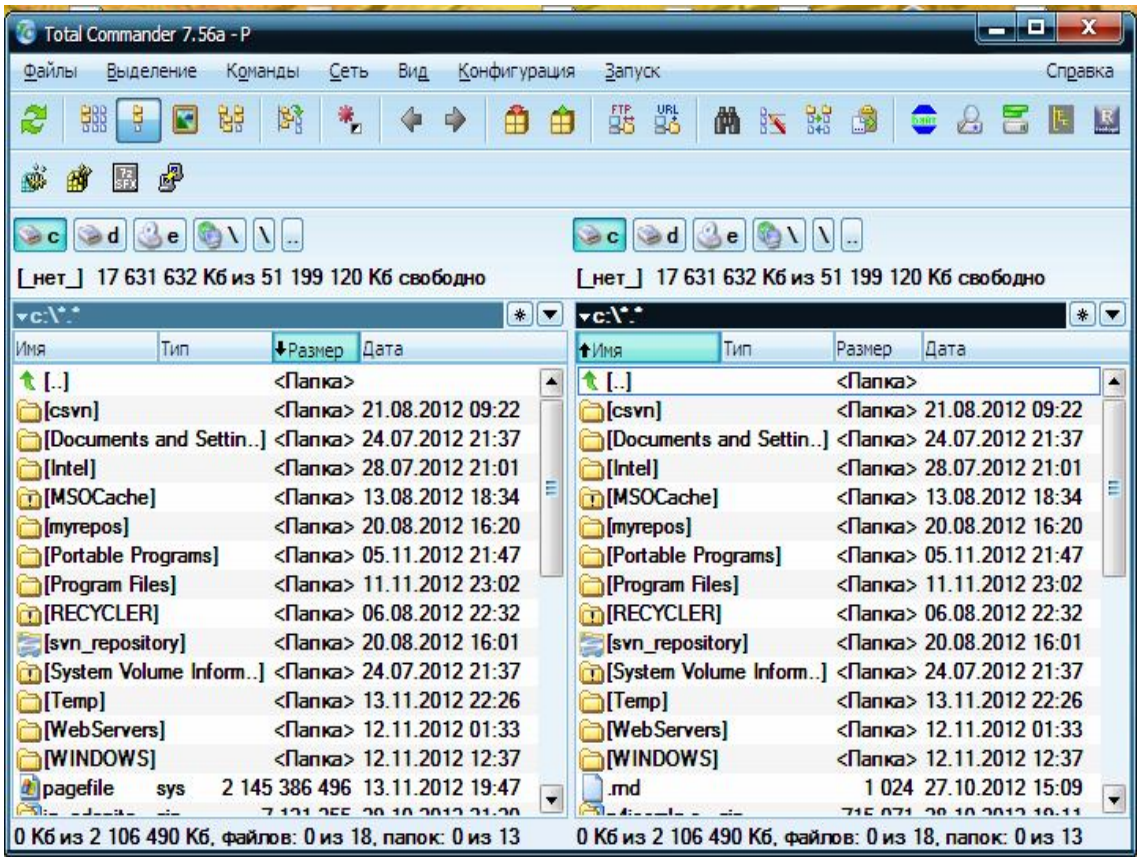

Рисунок 2.8 – Налаштуваня FTP у Total Commander

Після цього відкриється наступне вікно вибору FTP-серверів (рисунок 2.9):

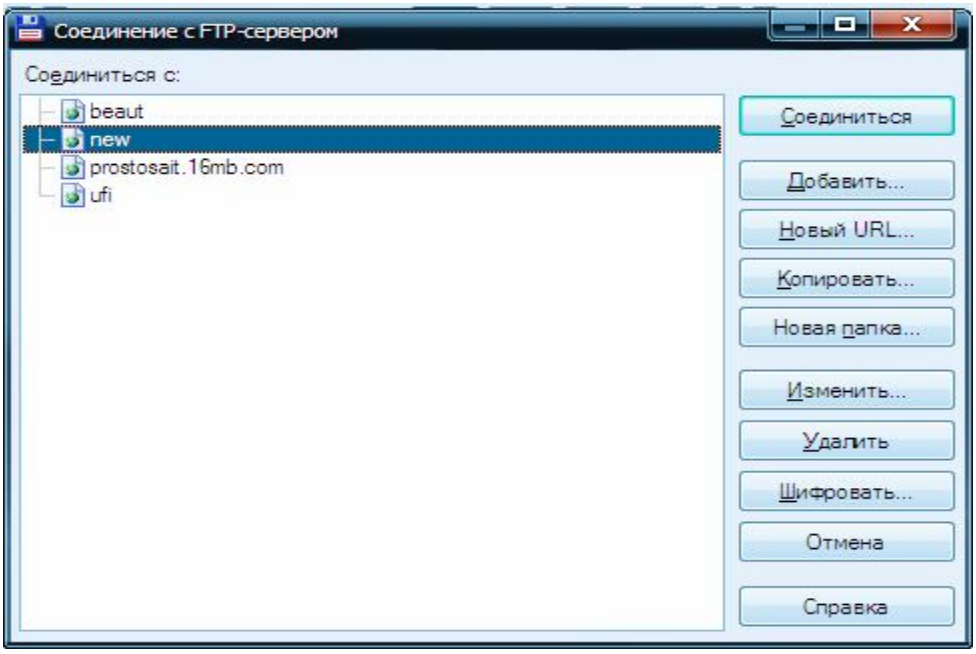

Рисунок 2.9 – Вікно з’єднання з FTP-сервером у Total Commander

Щоб створити новий FTP-сервер нажміть кнопку «Добавить» і побачите вікно «Настройка FTP-соединения» (рисунок 2.10):

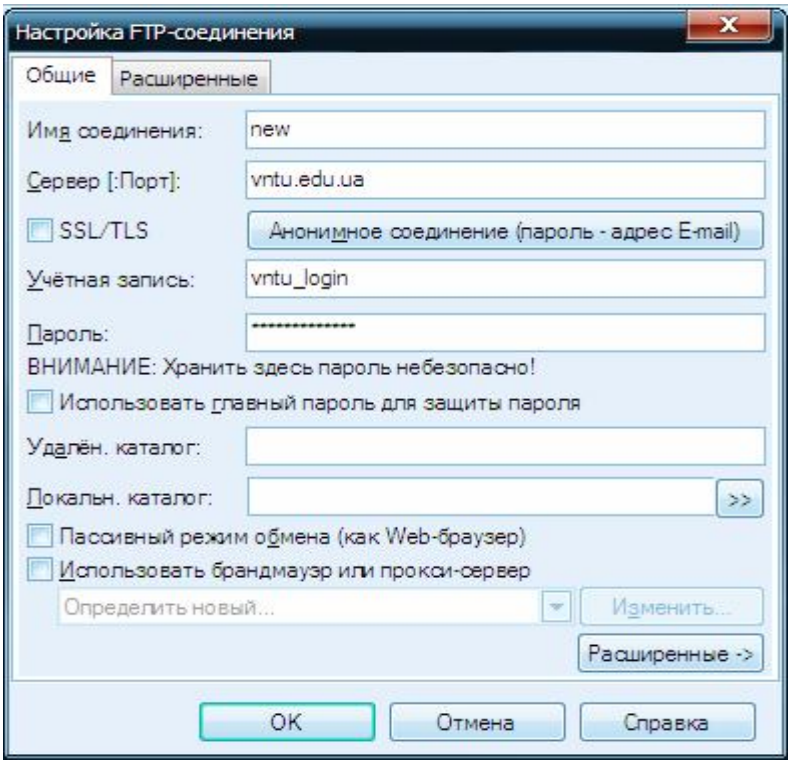

Рисунок 2.10 – Вікно налаштування FTP-з’єднання у Total Commander

У цьому вікні спочатку назвати з'єднання довільним ім’ям так, щоб ви потім розуміли, про яке саме з'єднання йде мова. Потім ввести ім'я FTP-

сервера, яке, зазвичай має вигляд ftp.mysyte.ru, але може бути і IP-адресою. Далі ввести FTP-логін і пароль. Після цього ввести найменування каталогу, в якому знаходиться ваш сайт. Зазвичай це public_html або www – скористайтеся отриманими у п.2 можливостями сайту hostinger.com.ua. Якщо ви ввели ім'я каталогу, то, з'єднавшись з сервером, Total Commander відразу перейде в цей каталог, і ви заощадите кілька секунд на самостійний перехід.

4.2 Far Manager. Запустити Far Manager. Відкрити список FTP-з'єднань, використовуючи клавіші ALT + F2 (вибрати "2: FTP" згідно рисунку 2.11).

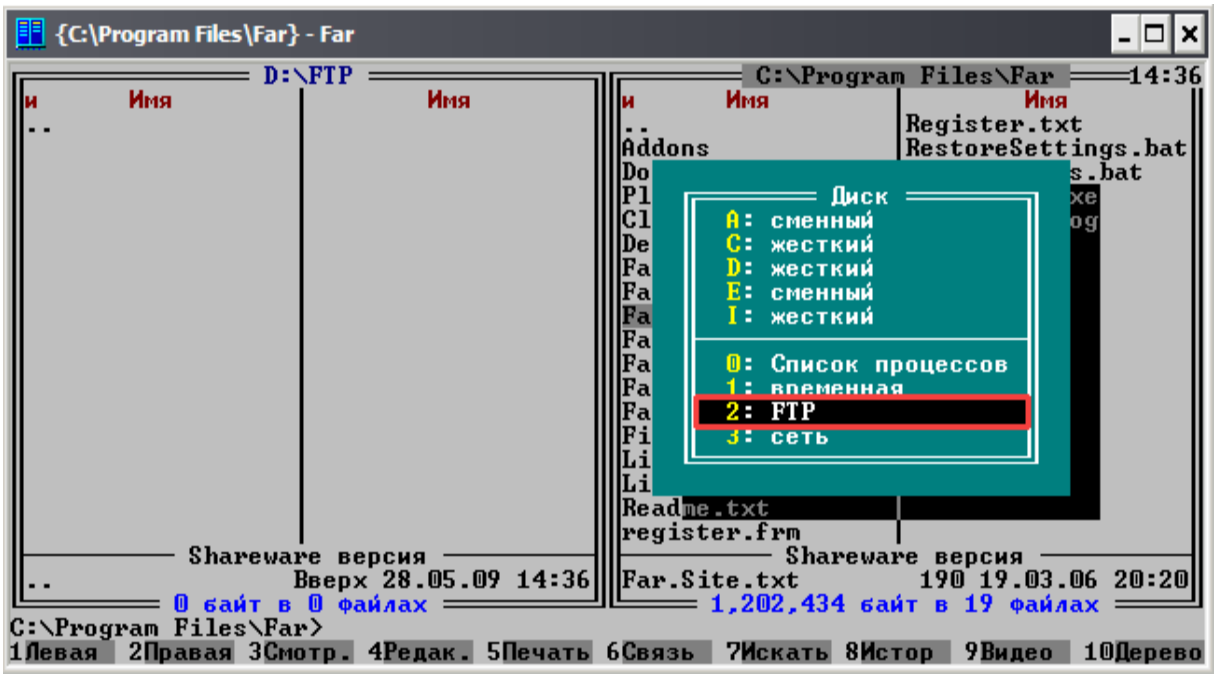

Рисунок 2.11 – Відкриття списку FTP-з’єднань у Far Manager

Якщо Ви вже створювали FTP-з'єднання, то зі списку з'єднань обрати потрібне та натиснути ENTER. Якщо ж Вам потрібно створити нове з'єднання, то натиснути клавіші SHIFT + F4.

Тепер заповнити основні поля згідно з рисунком 2.12:

- адреса FTP, Ім'я користувача, Пароль – уточнюються у хостера (власника сервера).
- обов'язково поставити галочку в пункті "Пасивний режим".

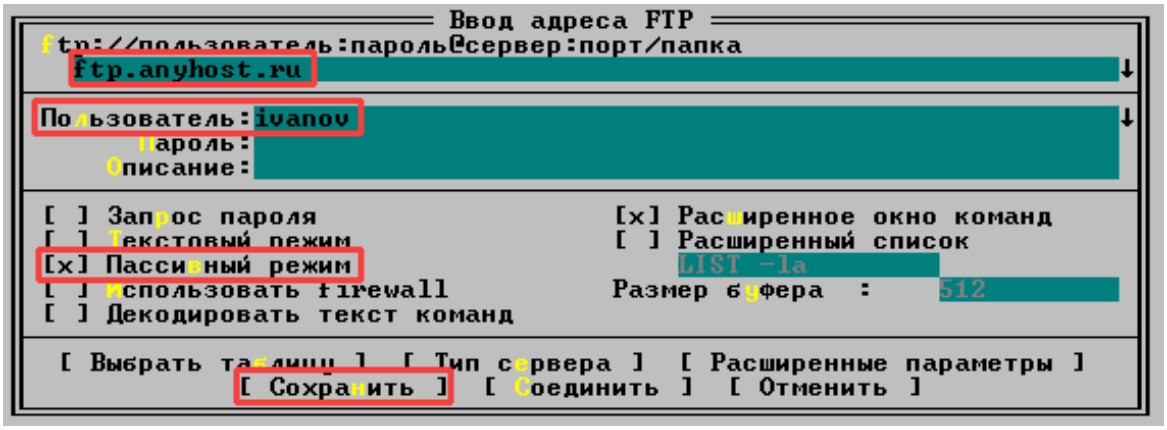

Рисунок 2.12 – Вікно налаштуваня FTP-з’єднання у Far Manager

Натисніть "Зберегти" - з'єднання створено. Тепер у списку з'єднань обрати потрібне з'єднання (яке Ви створили) і натиснути клавішу ENTER. Якщо всі дані вказані вірно, то відкриється вміст віддаленого сервера у вигляді файлів і папок (рисунок 2.13).

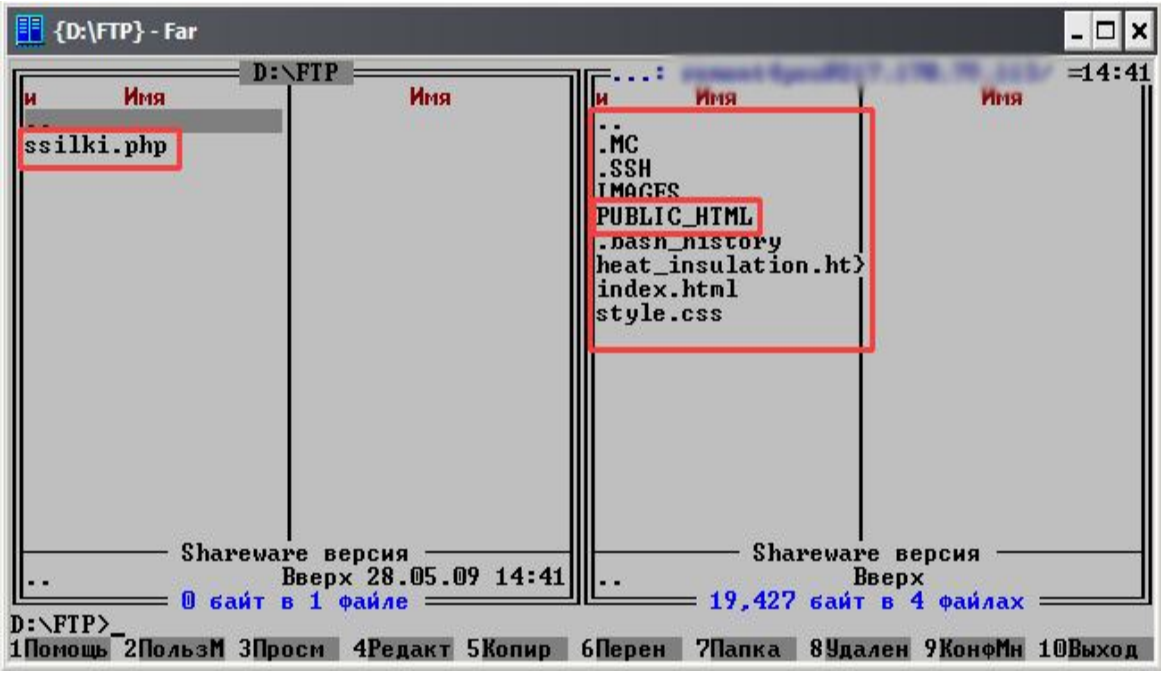

Рисунок 2.13 – Вміст віддаленого сервера у Total Commander

4.3 Провідник Windows. Відкрити "Мій комп'ютер". У рядку Адреса (де написано "Мій комп'ютер") ввести адресу FTP-доступу, який будується наступним чином: ftp://user:password@adress/ . Наприклад, у Вас є такі дані:

Адреса входу (adress): ftp.anyhost.ru

Логін (user): ivanov

Пароль (password): 123456

Тоді адреса FTP-доступу буде виглядати наступним чином: ftp://ivanov:123456 @ ftp.anyhost.ru / (рисунок 2.14).

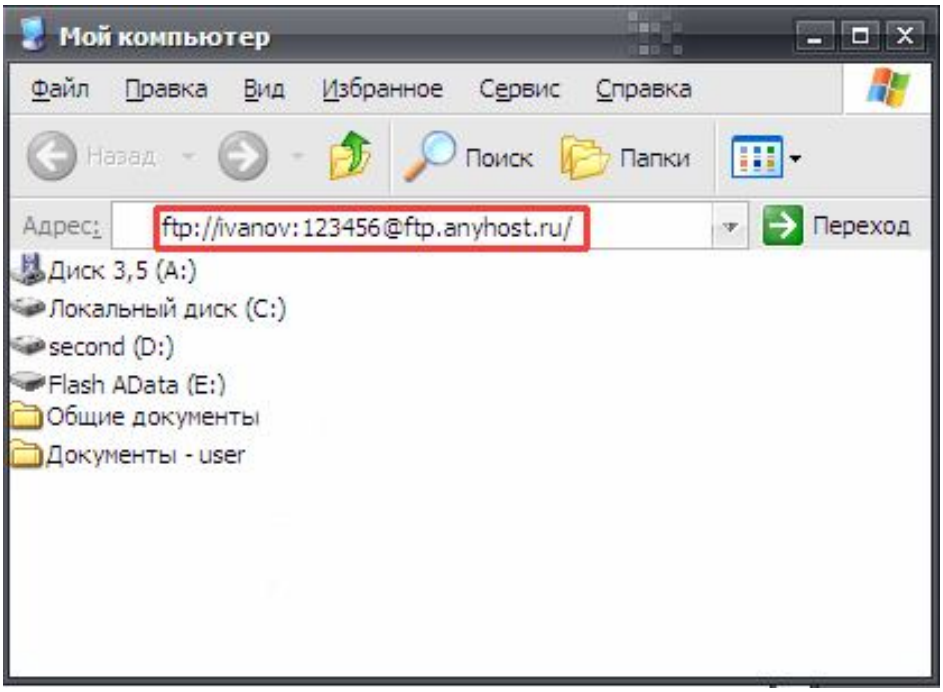

Рисунок 2.14 – Створення FTP-з’єднання в режимі «Мій комп'ютер»

Після того, як Ви ввели адресу FTP-доступу, натиснути "Перехід" або клавішу ENTER. Якщо всі дані вказані вірно, то відкриється вміст віддаленого сервера у вигляді файлів і папок (рисунок 2.15).

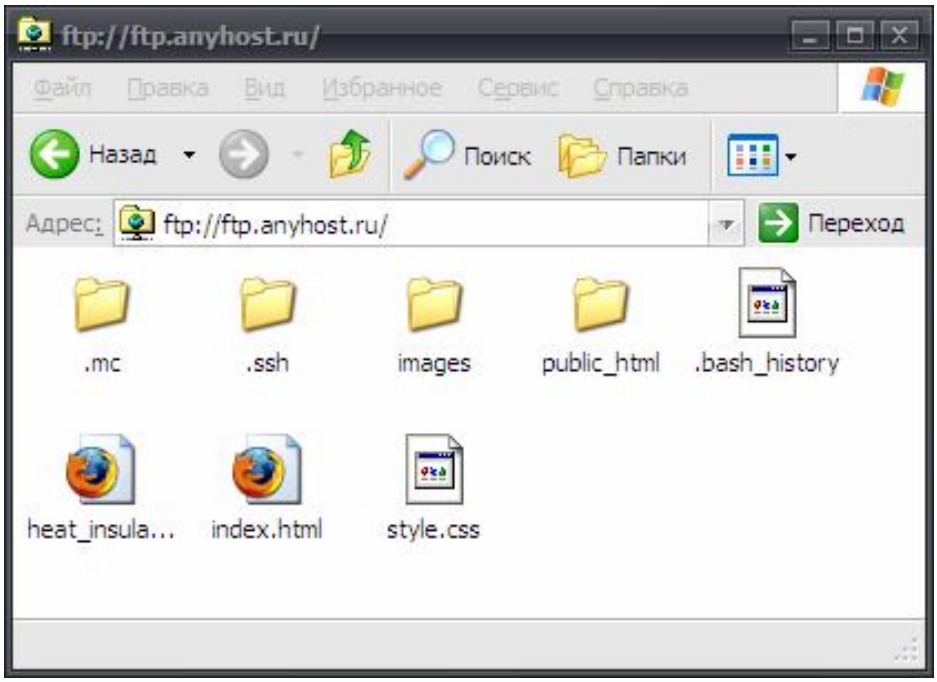

Рисунок 2.15 – Вміст віддаленого сервера у режимі «Мій комп'ютер»

4.4 Krusader**.** Шанувальники операційних систем Windows у повсякденній роботі досить часто використовують популярний файловий і FTP менеджер – Total Commander. Недивний той факт, що після переходу на операційну систему Lunux користувачі шукають аналог цієї зручної програми. Такий аналог дійсно існує та має назву Krusader. Функції його не тільки схожі з тими, що пропонує Total Commander, але й по багатьом параметрам Krusader перевершує свого конкурента.

Встановити за необхідності Krusader через «Центр додатків Ubuntu» (рисунок 2.16).

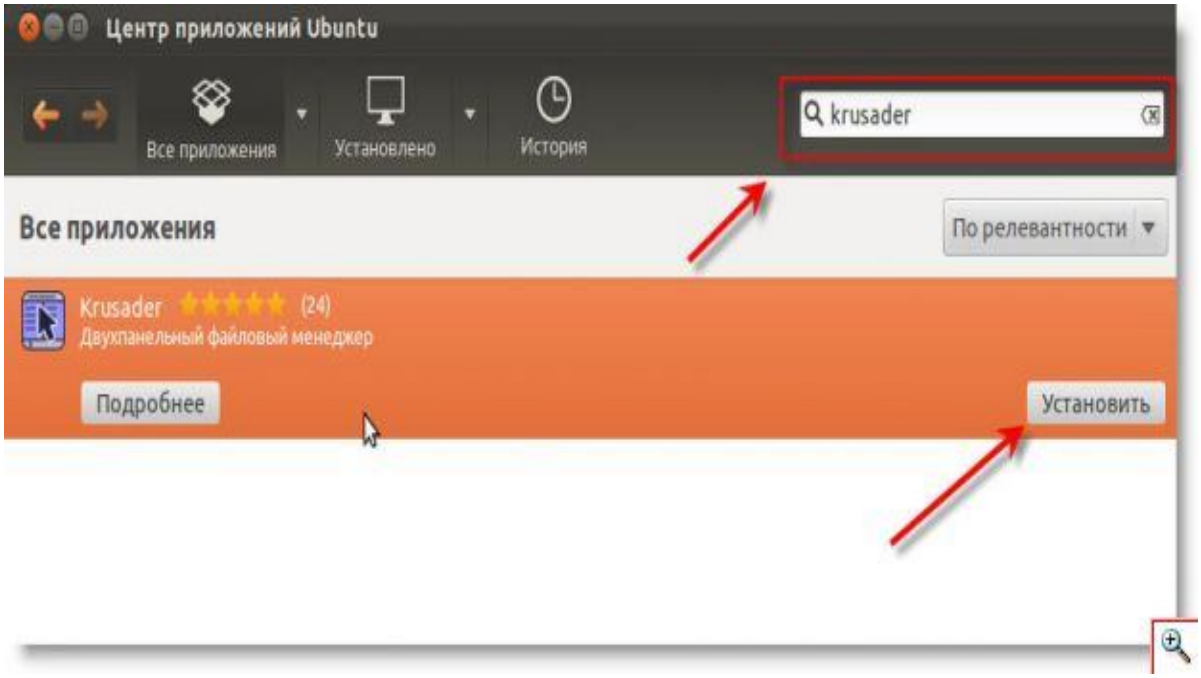

Рисунок 2.16 – Вміст віддаленого сервера у режимі «Мій комп'ютер»

Альтернативний метод для любителів командного рядка – ввести таку команду в терміналі: sudo apt-get install krusader.

Після установки знайти Krusader можна буде в «Головному меню» – «Інші програми» – «Стандартні». На вигляд Krusader (рисунок 2.17) є звичайним двохпанельним менеджером, що дозволяє копіювати, видаляти, вирізати, перейменовувати файли. Розглянемо налаштування FTP-доступу в Krusader.

Увійти в головне меню «Krusader»: «Сервис» – «Сетевое соединение». Відкриється нове вікно (рисунок 2.18), де необхідно послідовно заповнити поля:

Хост – адреса вашого FTP сервера;

Ім'я користувача – надається вашим провайдером ;

Пароль – надається вашим провайдером.

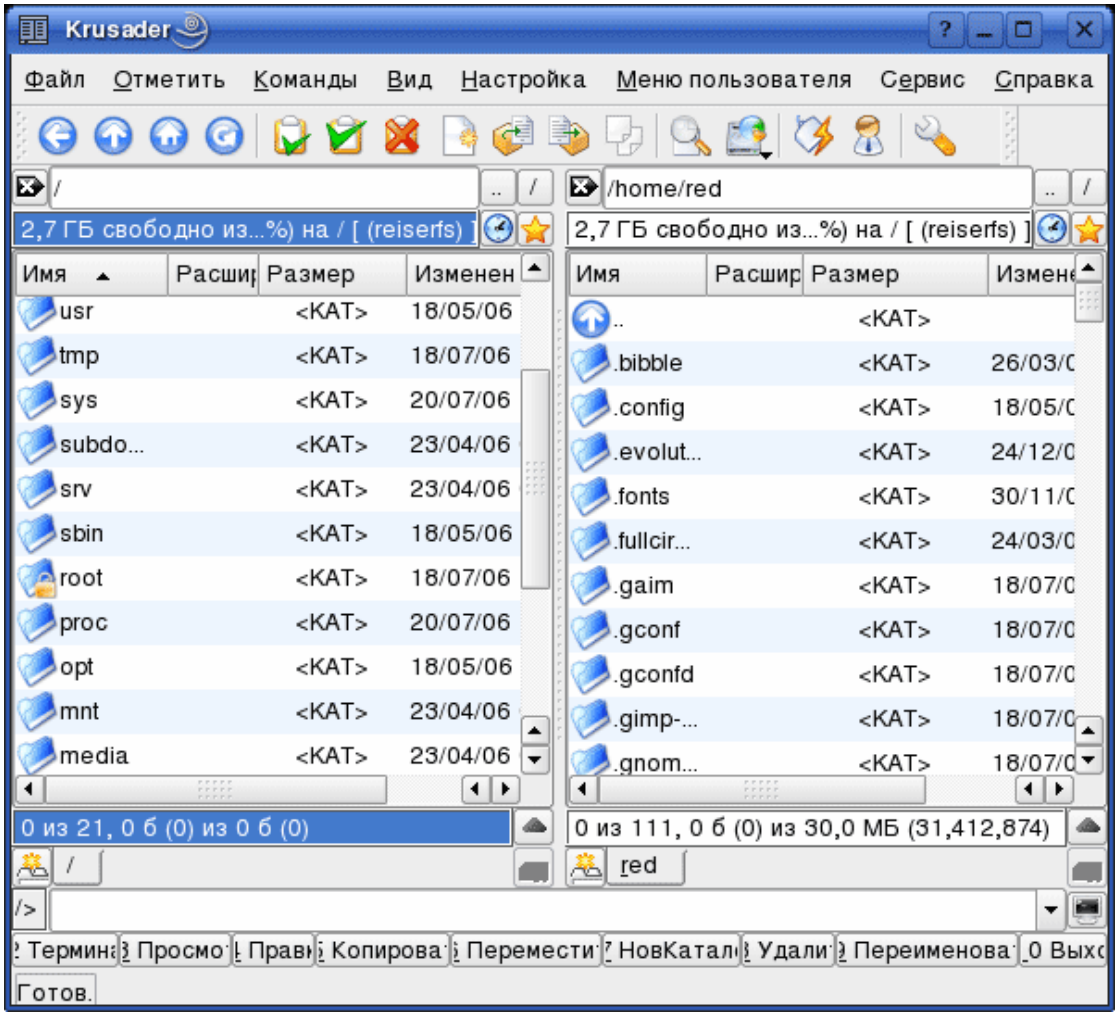

Рисунок 2.17 – Зовнішній вигляд менеджера Krusader

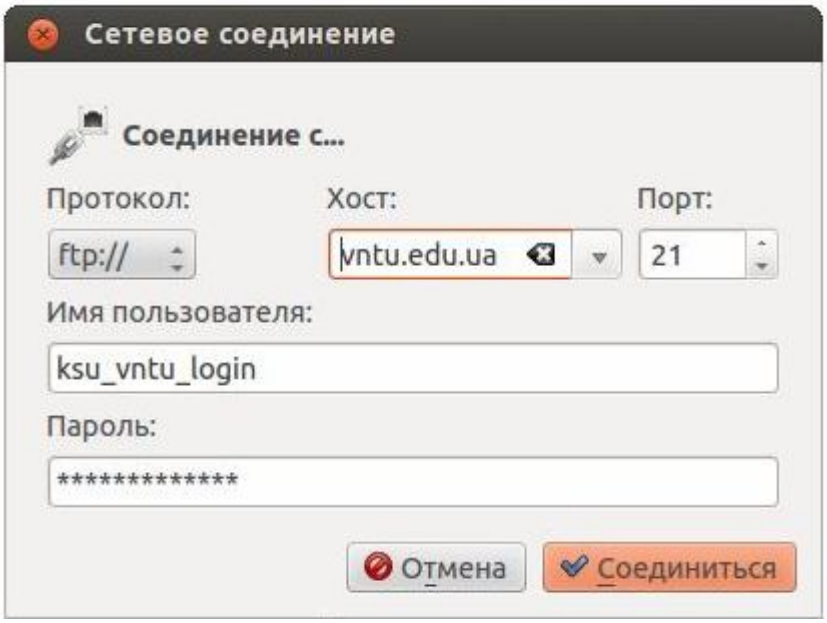

Рисунок 2.18 – Створення FTP-з’єднання в Krusader

Натиснути на кнопку «Соединиться» – якщо дані для доступу до вашого сервера введені вірно, то зачекати відкриття каталогів, розміщених на сервері та перевірити правильність з'єднання.

Вміст звіту

а) тема і мета лабораторної роботи;
б) у відповідності до кроків ходу роботи навести:
   1) графічне представлення інженерних дій у вигляді схем програм, схем взаємодії програмних модулів, діаграм використання або інших UML-діаграм;
   2) результати виконання всіх пунктів завдання лабораторної роботи;
в) змістовні висновки за результатами роботи.

Питання для самоконтролю

1. Що таке хостинг?
2. Які бувають умови надання хостингу?
3. Які існують види хостингу?
4. У чому відмінності варіантів повнофункціонального хостингу?
5. Що входить до послуг обмеженого хостингу?
6. Охарактеризуйте найбільш відомі безкоштовні хостинги?
7. Які основні служби забезпечує хостинг?
8. Що таке FTP?
9. Які переваги має FTP-клієнт у порівнянні з можливостями адмінок?
10. Які інструментальні засоби забезпечують альтернативні можливості створення FTP-клієнта?
11. Які параметри потрібні для налаштування FTP-з'єднання?
12. Основні можливості програми Krusader та її відмінності від інших FTP-менеджерів?

ЛАБОРАТОРНА РОБОТА №3. ОСНОВНІ КОНСТРУКЦІЇ МОВИ СЦЕНАРІЇВ PHP

Мета: ознайомлення з основами синтаксису PHP, одержання навичок у створюванні простих форм за допомогою PHP-редактора, дослідження операторів математичних операцій та введення даних у WEB-форму.

Теоретичні відомості

Скриптова мова (мова сценаріїв) програмування PHP, як правило, використовується суто для програмування додатків, пов'язаних з Інтернетом. PHP-сценарії вбудовуються у код HTML-сторінки, інтерпретуються і виконуються на сервері. Розглянемо приклад (рисунок 3.1): браузер звертається до WEB-сервера (наприклад, my_server) із запитом файла example.php з таким кодом (наприклад, http://my_server/example.php):

```
1   <html>
2   <head>
3   <title> example</title>
4   <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
5   </head>
6   <body>
7   Приклад простої PHP-сторінки <br>
8   Ці стрічки не інтерпретується, як код PHP <br> <br>
9   <?php
10  echo "Привіт, Я PHP-сценарій!";
11  ?>
12  <?
13  echo("<p>Це теж PHP-сценарій!<p>");
14  ?>
15  <?
16  echo("<p>Матеріал лабораторної роботи з дисципліни WEB-технології, підготував
17  В.В. Гармаш <p>");
18  ?>
19  </body>
20  </html>
```

Рисунок 3.1 – Приклад PHP-сторінки

Розширення файлу (*.php) для сервера є ознакою, що цей файл – скрипт і його треба виконати як програму за допомогою інтерпретатора PHP-коду. Виконавши за допомогою інтерпретатора цей PHP-скрипт, сервер одержить результат і надішле нашому браузеру таку WEB-сторінку:

Приклад простої PHP-сторінки

Ці стрічки не інтерпретується, як код PHP

Привіт, Я PHP-сценарій!

Це теж PHP-сценарій!

Матеріал лабораторної роботи з дисципліни WEB-технології, підготував В.В. Гармаш.

Розглянемо процес виконання php-сценарію при зверненні браузера до сервера. Отже, спочатку браузер запрошує у сервера сторінку з розширенням *.php. Одержавши такий запит, WEB-сервер виконує цей файл через інтерпретатор PHP і далі генерує результат у вигляді html-коду (згенерованої WEB-сторінки), який і надсилає у відповідь WEB-браузеру. Для того, щоб включити в файл команди PHP, необхідно використовувати спеціальні теги. В залежності від розширення файлу, їх розрізняють 4 види (вони еквівалентні і можна використовувати будь-які):

Інструкція обробки XML:

```
<?php
...
?>
```

Інструкція обробки SGML:

```
<?
...
?>
```

Інструкція обробки сценаріїв HTML:

```
<script language = "php">
...
</script>
```

Інструкція в стилі ASP:

```
<%
...
%>
```

У подальшому ми дотримуватимемося стилю XML або SGML.

Команда echo в PHP застосовується для виведення фактично всього, що зустрічається на web-сторінках (текст, розмітку HTML, числа). Значення її дії має бути зрозуміле з вищенаведеного прикладу.

Крапка з комою

Необхідно пам’ятати, що в кінці кожного рядку команди PHP-коду слід розмістити крапку з комою. Наприклад:

```
echo ("Тут крапка з комою ставиться після другої
фізичної стрічки, вказуючи на завершення команди. \n");
```

Але існує декілька виключень. Крапка з комою не ставиться, якщо у кінці рядку присутні:

- двокрапка (:);
- відкриваючі ({) та закриваючі (}) фігурні дужки;
- відкриваючі (<?) та закриваючі (?>) теги PHP.

Коментар – це фрагмент програмного коду, який не виконується і призначений лише для поточних пояснень і приміток. PHP надає декілька методів для вставки коментарів.

Перший спосіб пов'язаний з використанням подвійного слешу (//**).** Після цієї позначки PHP-інтерпретатор ігнорує все, що буде розташоване до кінця рядка.

Аналогічно можна користуватись і символом (#) – типовим способом коментування у скриптових мовах UNIX.

Для випадків, коли потрібно закоментувати одразу багато рядків вельми зручним стає в нагоді стиль мови програмування C (/*…*/) (рисунок 3.2).

```
1  <php
2  echo("<p>Hello</p>"); // коментар
3  echo("<p>Hello</p>"); # і це також коментар
4  $a = "Hello, world";
5  echo strstr($a,"H"); // цю функцію ми розглянемо пізніше
6  /*
7  і це теж коментарі
8  */
9  ?>
```

Рисунок 3.2 – Приклад коментарів в PHP

Слід пам'ятати про те, що стилі коментарів PHP діють тільки усередині обмежувачів PHP. Якщо PHP зустріне ці символи коментарів поза обмежувачами, то вони, як і будь-який текст, з'являться в на html-сторінці (рисунок 3.3).

```
1  <php
2  echo("<p>Hello</p>"); // правильно написаний коментар
3  ?> // а ось цей коментар буде відтворено браузером.
```

Рисунок 3.3 – Приклад коментарів поза обмежувачами в PHP

До речі, існує ще один коментар – HTML. Він не має безпосереднього відношення до PHP, але може бути корисним в HTML-сторінці. Відмітимо, що за своєю природою він є багаторядковим:

<-- Це - коментар HTML. Його можна побачити лише у вихідному коді HTML, але не у вікні WEB-браузера -->

У PHP змінні починаються зі знаку долара ($). За цим знаком може слідувати будь-яка кількість буквено-цифрових символів і символів підкреслення. Слід також пам'ятати, що у мові PHP назви змінних ЧУТливІ до РЕГістрУ.

При оголошенні змінних в PHP не вимагається явно вказувати тип змінної. При цьому одна і та ж змінна може впродовж програми набувати різних типів.

Змінна ініціалізується у момент привласнення їй значення і існує до тих пір, поки виконується програма (тобто скрипт). Це означає, що змінні існують до тих пір, поки не буде завершений запит (рисунок 3.4).

```
1  <?php
2  $string="Математичний приклад";
3  $a=2;
4  $b=3;
5  $num= $a+$b;
6  echo "$string <br> Сума дорівнює $num";
7  ?>
```

Рисунок 3.4 – Приклад оголошенні змінних в PHP

На рисунку 3.5 зображені арифметичні та логічні оператори.

Арифметичні оператори	
+	Додавання
-	Віднімання
*	Множення
/	Ділення
%	Обчислення залишку по модулю, наприклад: 5 % 2 = 1
Логічні оператори	
or або ||	АБО
Xor	Виключне АБО
and або +	І
!	Ні

Рисунок 3.5 – Арифметичні та логічні оператори

Робота з формами. Як відомо, HTML-форми використовуються для віддаленого введення інформації за допомогою WEB-браузера. В залежності від обставин, форми можуть мати різну кількість полів різного типу. PHP чудово взаємодіє з цими об'єктами і дозволяє конструювати потужні і зручні користувацькі інтерфейси. Більш того, власне синтаксис PHP дозволяє легко зчитувати дані з полів форми.

Але спочатку зазначимо, що будь-яка форма повинна вміщувати такі дескриптори й атрибути, які зображені на рисунку 3.6.

В атрибуті ACTION будемо задавати назву файлу з сценарієм (*.php), який має обробляти дані з HTML-форми.

```
1  <FORM NAME = «НАЗВА ФОРМИ»
2  ACTION = «ШЛЯХ ДО ПРОГРАМИ, ЩО ОБРОБЛЯЄ ДАНІ»
3  METHOD = «МЕТОД ПЕРЕДАЧІ ДАНИХ»>
4  ПОЛЯ ДЛЯ ВВЕДЕННЯ ІНФОРМАЦІЇ ….
5  </FORM>
```

Рисунок 3.6 – Синтаксис роботи з формами

Кожне поле введення інформації має атрибут NAME (наприклад, <INPUT TYPE=”text” NAME=”first” SIZE=”4” MAXLEGHT=”4”>), значення якого передається до програми, що має обробляти дані.

Для передачі даних із полів форми до програми використовується два методи: GET і POST. Під час використання методу GET після натискання кнопки на формі «Надіслати» значення полів автоматично приєднується до URL, вказаного в атрибуті ACTION: http://site.domain/ action.php?назв1 =знач1&...назв2=знач2

У випадку використання методу POST значення цих полів передаються в заголовку запиту до сервера, тобто непрозоро (непомітно) для користувача.

Хід роботи

1. Створити скрипт example_2.php, в якому послідовно перевірити всі демонстраційні приклади конструкцій PHP, наведені у теоретичних відомостях.

2. Комплексне завдання №1 – створення WEB-калькулятора**.**

2.1.Створити просту форму для проведення математичних дій над двома числами. У форму вмістити 2 поля для операндів та кнопку type="submit", за допомогою яких будуть надсилатись дані до WEB-сервера і дали оброблятись скриптом 2.php (рисунок 3.7).

```
1  <form action="2.php" method="get">
2  Ввести значення змінних: <br>
3  a: <input name="a" type="text" value=""><br>
4  b: <input name="b" type="text" value=""> <br>
5  <input type="submit" value="Розрахувати">
6  </form>
```

Рисунок 3.7 – Зразок створення простої форми для введення математичних дій над двома числами

2.2.Зробити форму більш привабливою за допомогою стандартних HTML-тегів (рисунок 3.8).

2.3.Зберегти файл з формою під назвою 1.php та, з метою перевірки, отримайте у браузері аналогічний рисунку 3.9 результат.

```
1   <html>
2   <head>
3   <title> Математичний приклад</title>
4   <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
5   </head>
6   <body>
7   <form ...
8   ……
9   </form>
10  </body>
11  </html>
```

Рисунок 3.8 – Стандартні HTML-теги

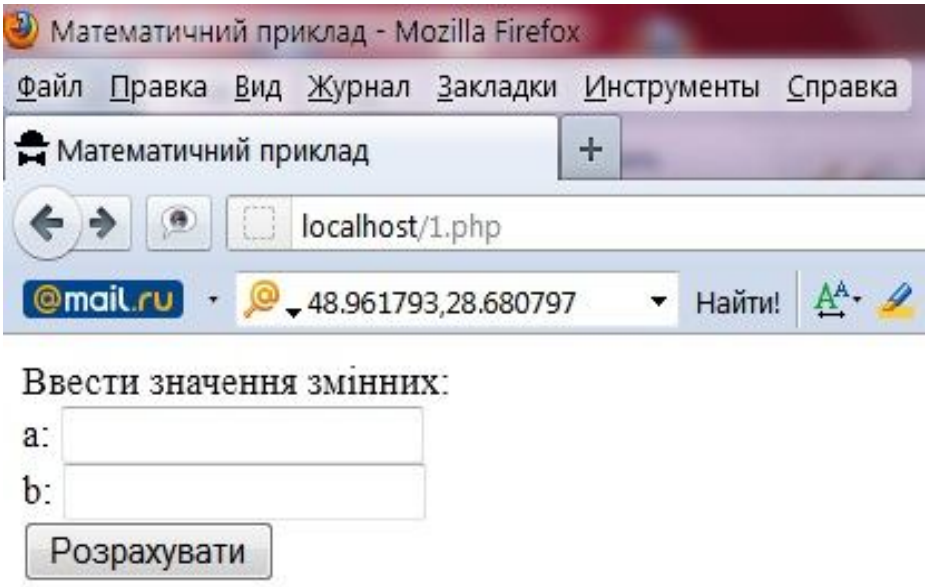

Рисунок 3.9 – Форма для введення даних у WEB-калькулятор

2.4. Створити PHP-скрипт, який буде отримувати з форми введені дані (змінні a і b) та виконувати над ними прості математичні дії (додавання / віднімання / множення / ділення). Для цього створюємо сценарій, в якому **м**іж PHP-тегами вставимо такий текст програми (рисунок 3.10).

2.5. Збережіть цей скрипт у новий файл під назвою 2.php.

2.6. Відкрийте браузер та в адресному рядку наберіть URL до файлу 1.php. Це має бути щось на кшталт: www.localhost/1.php.

2.7. Якщо ви все правильно зробили, то в браузері відкриється ваша форма. В ній введіть значення змінних a і b та натисніть кнопку «Розрахувати». Замість форми має з'явитися WEB-сторінка із результатами розра-

хунків (рисунок 3.11). Зверніть увагу на URL в браузері – це вже робота методу GET у скрипті 2.php.

```
1    <?php
2    $string="<font color=#FF0000><b>Математичний приклад</b></font>";
3    $a=$_GET[a];
4    $b=$_GET[b];
5    $sum= $a+$b;
6    $riz= $a-$b;
7    $dob= $a*$b;
8    $dil= $a/$b;
9    echo "$string<br><br> ";
10   echo "a=$a, b=$b<br><br> ";
11   echo "<b>Сума </b> $sum<br> ";
12   echo "<b>Різниця </b> $riz<br> ";
13   echo "<b>Добуток </b> $dob<br> ";
14   echo "<b>Частка </b> $dil<br> ";
15   ?>
```

Рисунок 3.10 – Вміст PHP-скрипта

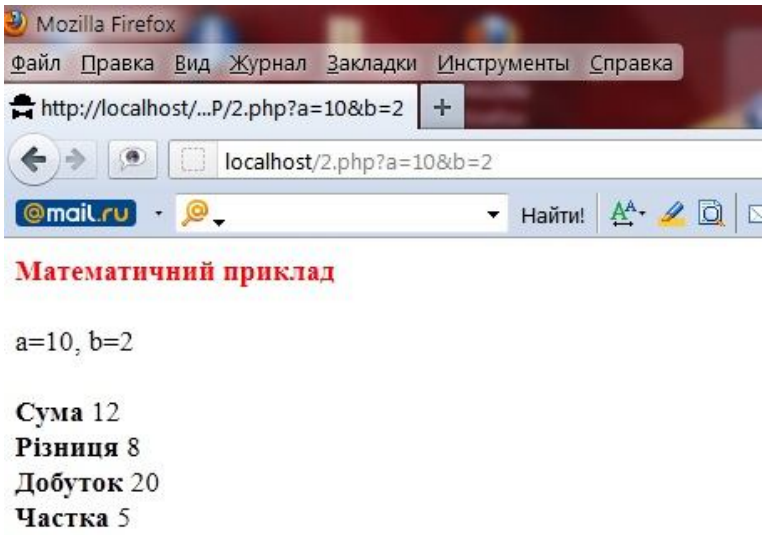

Рисунок 3.11 – Результат роботи скрипту 2.php

2.8. Перезаписати створені файли під іменами 11.php та 22.php, змінивши в них метод передачі даних з GET на POST. Порівняти отримані результати роботи цих двох методів та занотувати висновки до звіту.

3. Комплексне завдання №2 – створення WEB-форми для збирання та надсилання даних про користувача.

Припустимо, що нам потрібно за допомогою Internet провести опитування користувачів. Отримані результати маємо надсилати за певною адресою. Використовуючи набуті знання від попереднього завдання, можна

уявити, що і у цьому випадку треба також мати якусь WEB-форму, яку можна було би заповнювати з браузера, і яка могла би передавати свої дані відповідному скрипту для оброблення інформації.

У найпростішому випадку такий скрипт міг би просто генерувати WEB-сторінку із одержаними даними, як і у попередньому випадку. В інших випадках – він міг би надсилати одержані дані по електронній пошті, або заносити їх до бази даних.

Отже, початковий план – це генерація WEB-сторінки із рядками-гіперпосиланнями на інший скрипт. Зміст кожного гіперпосилання (URL) буде визначатись інформацією, яка буде занесена у першу WEB-сторінку з браузера. Загальний план виконання завдання №2 у вигляді схеми взаємодії програмних модулів представлено на рисунку 3.12.

3.1.Створити форму, за допомогою якої отримується анкетна інформація про користувача-студента (рисунок 3.13):

- навчальний заклад, в якому навчається студент;
- як можна з ним зв’язатись;
- захоплення студента.

3.2.Створити HTML-файл *about_you.htm*, який містить у собі цю форму з наступним змістом, як зображено на рисунку 3.14.

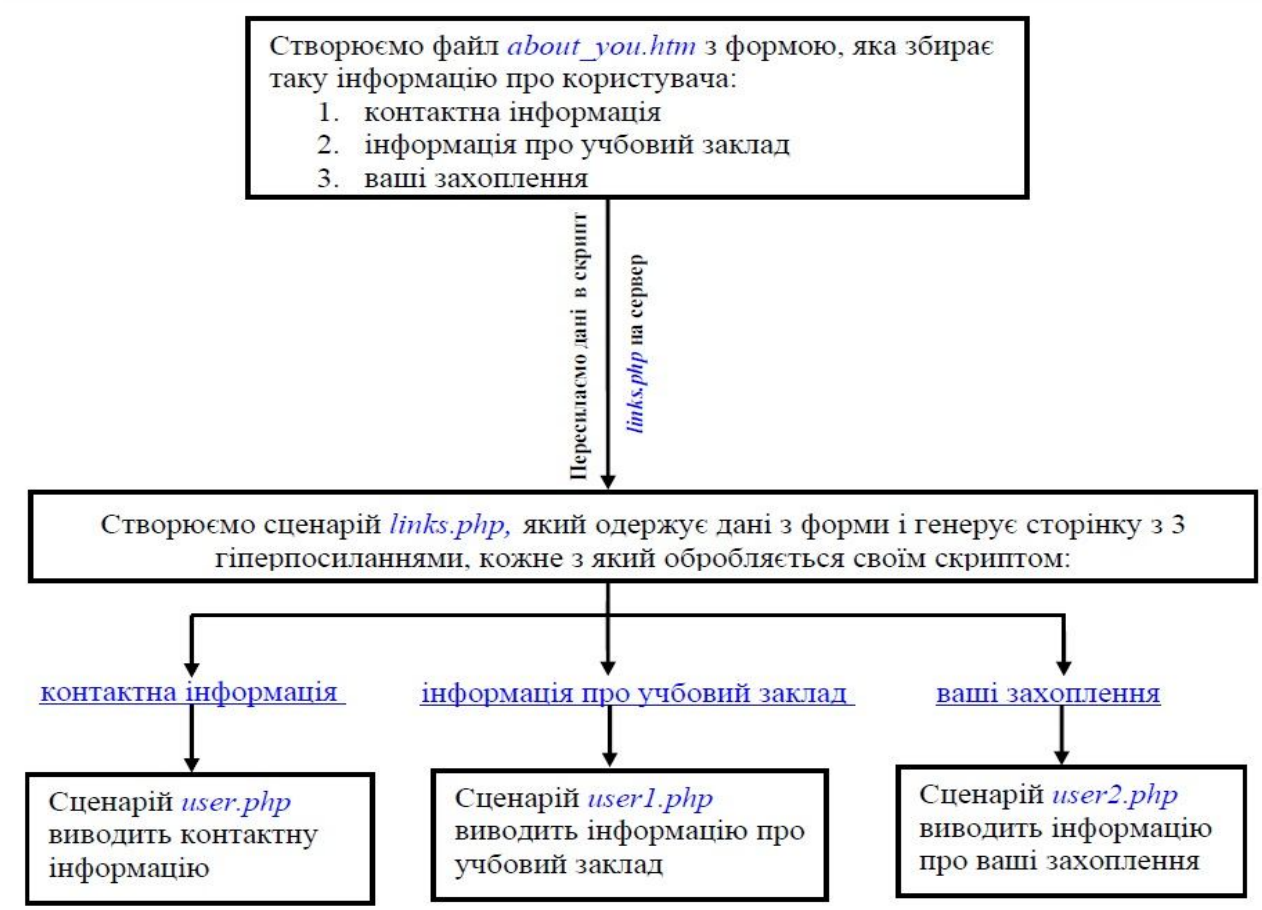

Рисунок 3.12 – Схема взаємодії програмних модулів для виконання завдання № 2

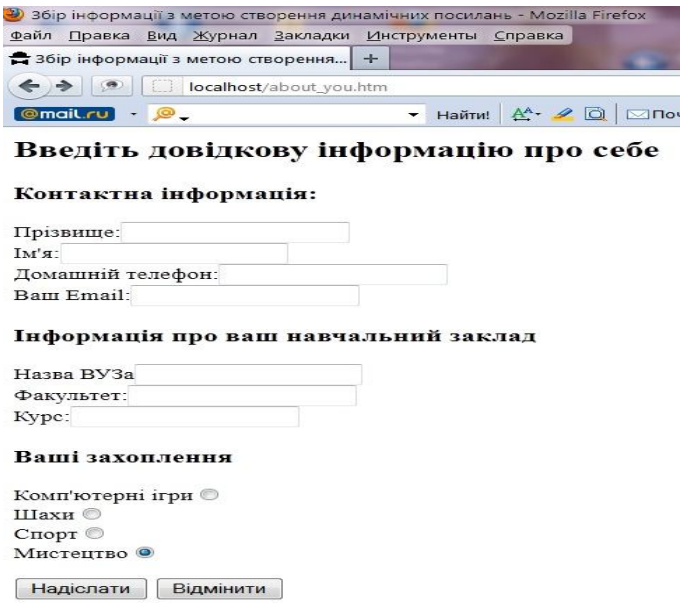

Рисунок 3.13 – Результат роботи сценарію *about_you.htm*

```
1   <html>
2   <head>
3   <title>Збір інформації з метою створення динамічних посилань</title>
4   </head>
5   <body>
6   <h2>Введіть довідкову інформацію про себе</h2>
7   <form action="links.php" method="GET">
8   <h3> Контактна інформація:</h3>
9   Прізвище:<input name="first_name" type="text">
10  <br>Ім'я:<input name="last_name" type="text">
11  <br>Домашній телефон:<input name="home_phone" type="text">
12  <br>Ваш Email:<input name="email" type="text">
13  <h3>Інформація про ваш навчальний заклад </h3>
14  Назва ВУЗа<input name="vuz_name" type="text">
15  <br>Факультет:<input name="fac_name" type="text">
16  <br>Курс:<input name="kurs" type="text">
17  <h3>Ваші захоплення </h3>
18  Комп'ютерні ігри<input name="hobby" type="radio"
19  value=" Комп'ютерні ігри" checked>
20  <br>Шахи<input name="hobby" type="radio" value="Шахи" checked>
21  <br>Спорт<input name="hobby" type="radio" value="Спорт" checked>
22  <br>Мистецтво<input name="hobby" type="radio" value=" Мистецтво" checked>
23  <p><input type="submit" name="Надіслати" value="Надіслати">
24  <input type="reset" name="Відмінити" value="Відмінити"></p>
25  </form>
26  </body>
27  </html>
```

Рисунок 3.14 – Вміст HTML-файл *about_you.htm*

3.3.Створити сценарій links.php, що буде одержувати дані з форми і створювати гіперпосилання, які далі можна застосувати для надсилання конкретних даних до іншого сценарію. Таких гіперпосилань у нас буде три. Як зображено на рисунку 3.15, перше з них призначене для отримання контактної інформації, друге – про навчальний заклад, в якому навчається студент, а третє – про захоплення користувача.

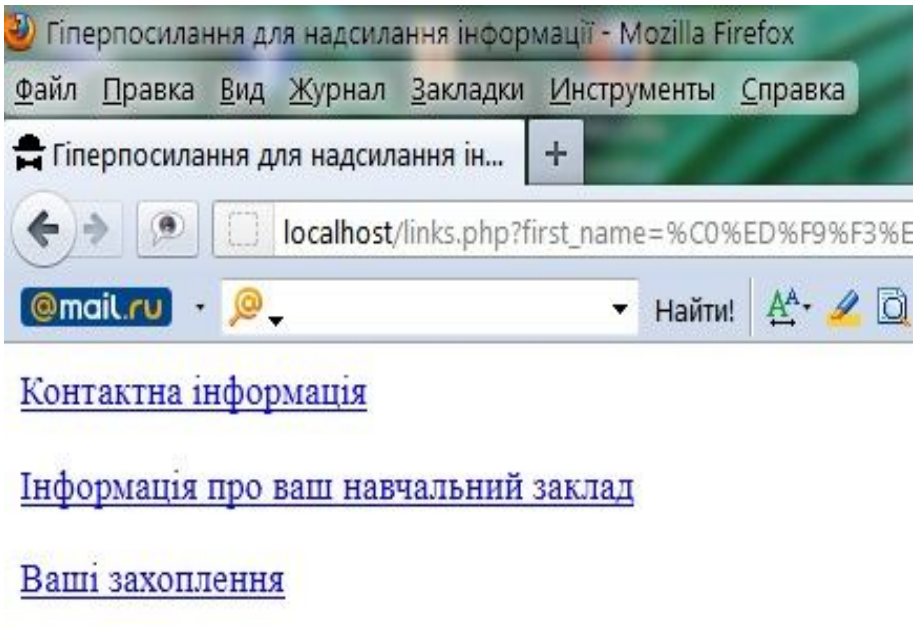

Рисунок 3.15 – Результат роботи сценарію *links.php*

3.4.Створити файл *links.php* з таким змістом, як зображено на рисунку 3.16.

Зверніть увагу, що у рядках 16–19 створюється гіперпосилання, яке передає контактну інформацію у сценарій *user.php*. Формуючи URL для сценарію *user.php*, потрібно додати пари «назва/значення» після назви сценарію для передачі йому даних.

Всі елементи, які знаходяться за знаком питання (?) в цьому URL, складаються із назви змінної та її значення, яке й передається з WEB-форми до сервера. Кожна із пар «назва/значення» відокремлена від іншої символом амперсанд (&).

У рядках 20–22 аналогічно створюємо гіперпосилання, яке передає інформацію про учбовий заклад у сценарій *user1.php*; і у рядках 23–24 створюємо посилання про ваші захоплення, яке передає інформацію у сценарій *user2.php*.

3.5.Створити сценарій user.php, який має виводити дані про користувача, якщо у сценарії *links.php* було обрано гіперпосилання “Контактна інформація” (рисунок 3.17).

```
1   <html>
2   <head>
3   <title>Гіперпосилання для надсилання інформації</title>
4   </head>
5   <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
6   <body>
7   <?php
8   $first_name=$_GET[first_name];
9   $last_name=$_GET[last_name];
10  $home_phone=$_GET[home_phone];
11  $email=$_GET[email];
12  $vuz_name=$_GET[vuz_name];
13  $fac_name=$_GET[fac_name];
14  $kurs=$_GET[kurs];
15  $hobby=$_GET[hobby];
16  echo "<p><a
17  href=\"user.php?first_name=$first_name&last_name=$last_name
18  &home_phone=$home_phone&email=$email\">";
19  echo "Контактна інформація</a></p>";
20  echo "<p><a
21  href=\"user1.php?vuz_name=$vuz_name&fac_name=$fac_name&kurs=$kurs\">";
22  echo "Інформація про ваш навчальний заклад</a></p>";
23  echo "<p><a href=\"user2.php?hobby=$hobby\">";
24  echo "Ваші захоплення</a></p>";
25  ?>
26  </body>
27  </html>
```

Рисунок 3.16 – Вміст файлу *links.php*

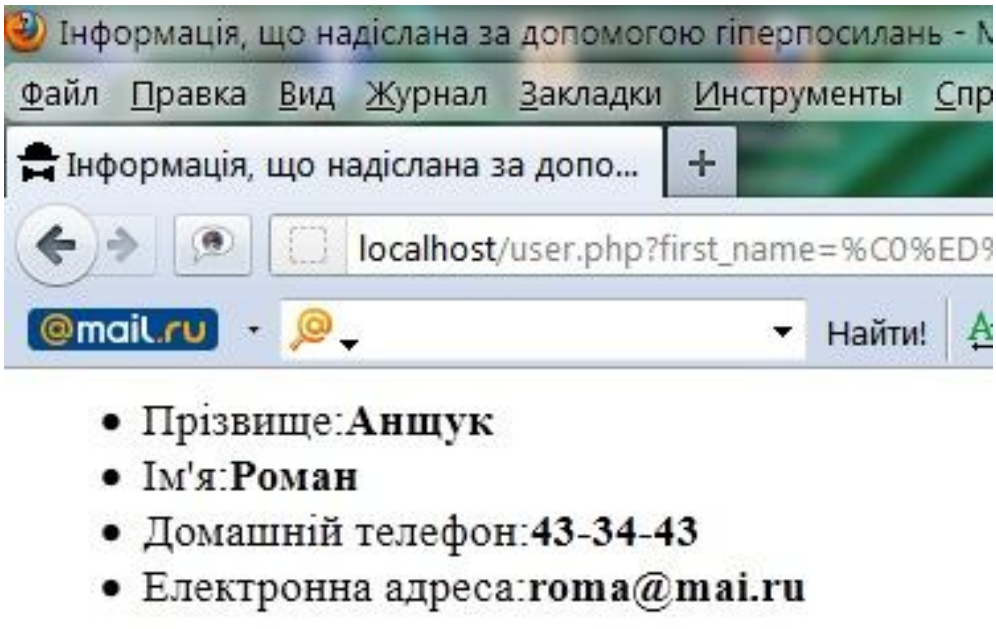

Рисунок 3.17 – Результат роботи сценарію *user.php*

Зміст скрипта має бути таким, як зображено на рисунку 3.18.

```
<html>
<head>
<title>Інформація, що надіслана за допомогою гіперпосилань</title>
</head>
<body>
<?php
  $first_name=$_GET[first_name];
  $last_name=$_GET[last_name];
  $home_phone=$_GET[home_phone];
  $email=$_GET[email];
?>
<ul>
<li> Прізвище:<b><?php echo $first_name ?></b></li>
<li> Ім'я:<b><?php echo $last_name ?></b></li>
<li> Домашній телефон:<b><?php echo $home_phone ?></b></li>
<li> Електронна адреса:<b><?php echo $email ?></b></li>
</ul>
</body>
</html>
```

Рисунок 3.18 – Вміст скрипта

3.6. Самостійно створити сценарії *user1.php* та *user2.php* у відповідності до схеми взаємодії програмних модулів на рисунку. 3.12.

Вміст звіту:

а) тема і мета лабораторної роботи;
б) у відповідності до кроків ходу роботи навести:
   1) графічне представлення інженерних дій у вигляді схем програм, схем взаємодії програмних модулів, діаграм використання або інших UML-діаграм;
   2) лістинг всіх відлагоджених програмних модулів;
   3) результати виконання програмних модулів;
в) змістовні висновки за результатами роботи.

Питання для самоконтролю:

1. Де виконуються PHP-сценарії?
2. Яким чином PHP-сценарій вбудовується в HTML-сторінку?
3. Яка команда використовується для виведення інформації на екран?
4. Що ставиться в кінці кожної стрічки команди PHP-коду?
5. Коли не ставиться крапка з комою?
6. Як можна вставити коментарі на сторінку?
7. Як позначаються змінні у PHP-сценарії?

8. Для чого існує атрибут ACTION у формі?
9. Які є методи передачі даних з форми і як вони працюють?
10. Куди передаються дані з форми і як вони обробляються?
11. Яким чином додається назва і значення змінної до URL?
12. Яким чином формується гіперпосилання у скрипті *links.php*?
13. Які додаткові скрипти використовуються у сценарії *links.php* і для чого?
14. Які елементи знаходяться за знаком (?) в URL, коли дані передаються з форми до сервера?
15. Що означає знак & (амперсанд) в URL, коли дані передаються з форми до сервера?
16. Для чого потрібен скрипт *user.php*? Яким чином передається інформація у цей сценарій?
17. Для чого потрібен скрипт *user1.php*? Яким чином передається інформація у цей сценарій?
18. Для чого потрібен скрипт *user2.php*? Яким чином передається інформація у цей сценарій?

ЛАБОРАТОРНА РОБОТА №4. ОПЕРАТОРИ УМОВНОГО ПЕРЕХОДУ ТА ЦИКЛІВ У PHP

Мета: набуття навичок у використанні конструкцій розгалуження та циклічних дій мови PHP.

Теоретичні відомості

До операторів умовного переходу відносять: оператор (if...else) і перемикач (switch). Синтаксис умовного оператора у PHP такий:

if (condition) statement_1 else statement_2

Умова *condition* може бути будь-яким виразом. Якщо вона є істиною, то виконується оператор *statement_1*. Інакше виконується оператор *statement_2*. Допускається скорочена форма запису умовного оператора, в якій відсутні оператор *else* і оператор *statement 2*.

У свою чергу, оператори *statement_1*і *statement_2* можуть бути умовними, що дозволяє організовувати ланцюжки перевірок будь-якої глибини вкладеності. І в цих ланцюжках кожний умовний оператор може бути як повним, так і скороченим.

Синтаксис мови PHP припускає, що при вкладених умовних операторах кожне *else* відповідає найближчому *if*. У зв'язку з цим можливі помилки неоднозначного зіставлення *if* та *else*. Простим правильним рішенням цієї задачі є застосування фігурних дужок, тобто нам потрібно фігурними дужками обмежити область дії внутрішнього умовного оператора, зробивши його неповним. Тим самим зовнішній оператор перетворюється на повний умовний (рисунок 4.1).

```
1   <?
2   $x = 1;
3   $y = 1;
4   if($x==1)
5   {
6   if($y==1)echo("x=1 and y=1");
7   }
8   else echo("x!=1");
9   ?>
```

Рисунок 4.1 – Приклад застосування операторів умовного переходу

Додаткові умови можливо перевірити за допомогою оператора elseif. Оператор if може включати скільки завгодно блоків elseif, але else в кожному if може бути тільки один. Як правило, в конструкціях if…elseif…else

оператор else визначає, що потрібно робити, якщо ніякі інші умови не виконуються.

Розглянемо приклад – магазин надає знижки при замовленні великої кількості автопокришок. Схема знижок виглядає таким чином:

- придбання менше 10 автопокришок – без знижки;
- придбання 10-49 автопокришок – знижка 5%;
- придбання 50-99 автопокришок – знижка 10%;
- придбання 100 і більше автопокришок – знижка 15%.

Можна створити код для обчислення знижок з використанням умов і операторів if і elseif (рисунок 4.2). Для об'єднання двох умов в одне застосовується операція і (&&). Значення змінної *$tireqty* вводиться за допомогою html-форми (розробити таку самостійно).

```
1   <?php
2   $z="<b>Знижка дорівнює</b>";
3   $n="<b>Без знижки </b>";
4   If  ( $tireqty >= 10 && $tireqty <= 49 ) {
5   $discount = 5;
6   echo "$z  $discount<br> ";}
7   elseif ( $tireqty >= 50 && $tireqty <= 99 ) {
8   $discount = 10;
9   echo "$z  $discount<br> ";}
10  elseif( $tireqty >= 100 ){
11  $discount = 15;
12  echo "$z  $discount<br> ";  }
13  else
14  echo "$n";
15  ?>
```

Рисунок 4.2 – Приклад програмного коду з використанням умов і операторів if та elseif

Зверніть увагу на те, що можна застосовувати як elseif, так і else if - обидва варіанти правильні.

При використанні каскадних наборів операторів elseif слід пам'ятати, що буде виконуватися тільки один з блоків операторів. У даному прикладі це не важливо оскільки всі умови є взаємовиключними - в кожен момент часу може виконуватися тільки один з них. Якби умови були записані так, що одночасно могло б виконуватися декілька умов, виконувався б тільки блок, або оператор, наступний за першою дійсною умовою.

Проте, використання оператора elseif досить сильно погіршує читабельність коду, і краще в цьому випадку користуватись перемикачем switch, який призначений для аналізу множинних умов.

PHP надає також можливість альтернативного синтаксису умовного оператора – без фігурних дужок – із застосуванням оператора endif. Наприклад :

if(вираз): блок_виконання endif;

Має сенс, якщо умова, записана в круглих дужках оператора if, виявилася істиною, виконуватиметься ваш код, від двокрапки «:» до команди endif. Використання такого синтаксису корисне при вбудовуванні php в html-код (рисунок 4.3).

```
1  <?php
2  $names = array("Іван","Петр","Семен");
3  if ($names[0]=="Іван"):
4  ?>
5  Привіт, Ваня!
6  <?php endif ?>
```

Рисунок 4.3 – Приклад застосування альтернативного синтаксису умовного оператора

У наступному прикладі скрипт аналізує зміст змінної $HDD. Якщо в ній буде стрічка «Maxtor», то буде генеруватись WEB-сторінка з таблицею і заголовком «Maxtor». У протилежному випадку – це буде таблиця із заголовком «Seagate». Наявність оператора endif в цьому випадку обов'язкова, оскільки фігурна дужка, що позначає кінець блоку if, відсутня
(рисунок 4.4).

```
<?
  if($HDD == «Maxtor»):
?>
<table>
   <caption> Maxtor </caption>
</table>
<?
  elseif($HDD == «Seagate»):
?>
<table>
  <caption> Seagate </caption>
</table>
<?
  endif;
?>
```

Рисунок 4.4 – Аналіз змісту змінної $HDD

Результати виконання скрипта будуть такими:
а) форма запиту (рисунок 4.5):

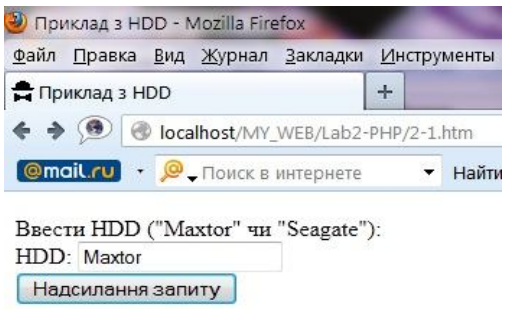

Рисунок 4.5 – Вигляд форми запиту 1 у браузері

б) результат виконання (рисунок 4.6):

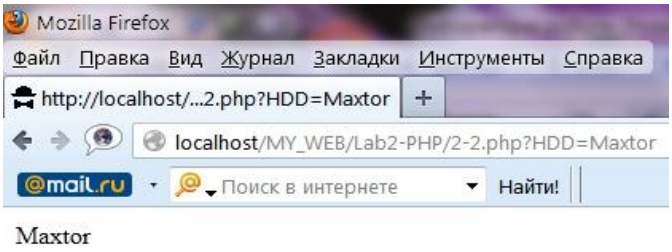

Рисунок 4.6 – Результат виконання запиту 1 у браузері

PHP надає можливість замінювати блоки if…else умовною операцією. У зображенні умовної операції присутні два розміщених не підряд символи ‘?’ і ‘:’ і три операнди виразу:

вираз_1 ? вираз_2 : вираз_3

Першим обчислюється значення виразу_1. Якщо воно є істинним (тобто не дорівнює нулю), то обчислюється значення виразу_2, яке і стає результатом. Якщо при обчисленні значення вираз_1 буде дорівнювати нулю (хибність), то результат буде дорівнювати виразу_3 (рисунок 4.7).

```
1   <?
2   if($HDD == "Maxtor")
3   {
4   $CDROM = "Teac";
5   }
6   else
7   {
8   $CDROM = "Nec";
9   }
10  ?>
```

Рисунок 4.7 – Приклад застосування умовних операцій

Код, який зображений на рисунку 4.7 можна записати таким чином (рисунок 4.8).

```
1    <?
2    $CDROM = ($HDD == "Maxtor") ? "Teac" : "Nec";
3    ?>
```

Рисунок 4.8 – Один з варіантів застосування умовних операцій

Перемикач switch є найзручнішим засобом для організації розгалуження із множинними варіантами умов (рисунок 4.9).

```
1    switch(expression) // перемикаючий вираз
2    {
3    case value1: // константний вираз 1
4    statements; // блок операторів
5    break;
6    case value2: // константний вираз 2
7    statements;
8    break;
9    default:     // інакше ...
10   statements;
11   }
```

Рисунок 4.9 – Синтаксис перемикача

Структура керування switch передає управління тому з помічених case операторів, для якого значення константного виразу співпадає із значенням перемикаючого виразу. Якщо значення перемикаючого виразу не співпадає ні з одним з константних виразів, то виконується перехід до оператора, поміченого міткою default. У кожному перемикачі може бути не більше однієї мітки default (проте вона може бути і взагалі відсутньою).

Нижче, на рисунку 4.10 наведемо приклад програми з перемикачем. У цій програмі виводяться назви непарних цілих десяткових цифр від 1 до 9 не менше заданого, залежно від числа, вказаного у формі test.html. Форма test.html не відрізняється від тієї, що ми вже використовували.

Результати виконання скрипта будуть такими:

а) форма запиту (рисунок 4.11);

б) результат виконання (рисунок 4.12).

Для перемикачів допустимі будь-які ступені вкладеності, проте зловживати цим без особливої на те необхідності також не слід.

Приведена програма демонструє дію оператора break, за допомогою якого відбувається вихід з перемикача. Якщо помістити оператори break після виведення кожної з цифр, то у вікні браузера ми побачимо назву тільки однієї непарної цифри.

```
1    <?
2    switch($number)
3    {
```

```
4    case 1:
5    echo ("one ");
6    case 2: case 3:
7    echo ("three");
8    case 4: case 5:
9    echo ("five");
10   case 6: case 7:
11   echo ("seven");
12   case 8: case 9:
13   echo ("nine");
14   break;
15   default:
16   echo ("Ці номери не потрапляють у заданий діапазон > 9 or < 1");
17   }
18   ?>
```

Рисунок 4.10 – Приклад програми з перемикачем

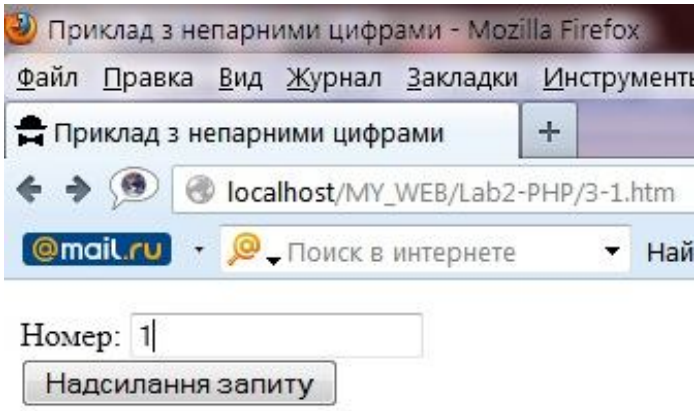

Рисунок 4.11 – Вигляд форми запиту 2 у браузері

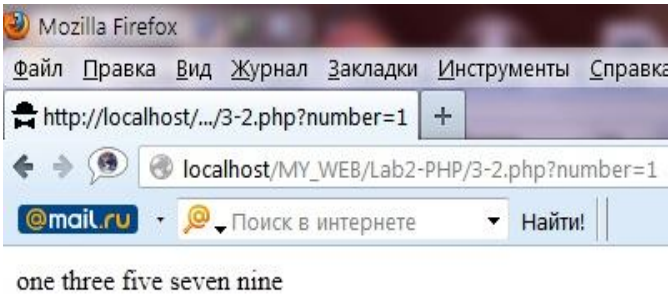

Рисунок 4.12 – Результат виконання запиту 2 у браузері

Оператори циклу. Мова PHP підтримує 4 типи циклів:

- цикл з передумовою:

```
while(condition) {statements;}
```

- цикл з постумовою:

```
do{statements;} while(condition);
```

- ітераційний цикл:

```
for(expression1;expression2;expression3)
{statements;}
```

- ітераційний цикл foreach:

```
foreach (array as [$key =>] $value)
{statements;}
```

Оператор циклу While. Оператор while називається оператором циклу з передумовою. При вході в цикл обчислюється умова, і, якщо її значення відмінне від нуля, виконується тіло циклу. Потім обчислення умови і операторів тіла циклу виконується до тих пір, поки значення виразу умови не стане рівним нулю (рисунок 4.13). Оператором while зручно користуватися для проглядання довільних послідовностей, якщо в кінці їх знаходиться наперед відомий символ.

```
1   <?
2   $var = 5;
3   $i = 0;
4   while(++$i <= $var)
5   {
6   echo($i); echo('<br>');
7   }
8   ?>
```

Рисунок 4.13 – Приклад простого циклу while

Код, який зображений вище видає у вікні браузера цифри від одного до п'яти (рисунок 4.14):

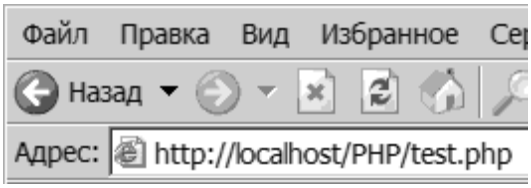

Рисунок 4.14 – Результат виконання прикладу для циклу while

Для виходу з циклу застосовується оператор break. При виявленні цього оператора поточна ітерація циклу припиняється, і подальші ітерації не відбуваються (рисунок 4.15). При виконанні наступного прикладу, не зва-

жаючи на те, що змінна $var = 7, у вікні браузера з'являться цифри від 1 до 5.

```
1   <?
2   $var = 7;
3   $i = 0;
4   while(++$i <= $var)
5   {
6   echo($i);
7   echo('<br>');
8   if($i==3)break;
9   }
10  ?>
```

Рисунок 4.15 – Приклад застосовування оператора break

Іноді буває потрібно перервати тільки поточну ітерацію, і перейти відразу до наступної. Для цього застосовується оператор continue (рисунок 4.16).

```
1   <?
2   $var = 7;
3   $i = 0;
4   while(++$i <= $var)
5   {
6   if($i==5)
7   {
8   continue;
9   }
10  echo($i);
11  echo('<br>');
12  }
13  ?>
```

Рисунок 4.16 – Приклад застосовування оператора continue

В даному прикладі виводяться цифри від 1 до 7, окрім цифри 5 (рисунок 4.17):

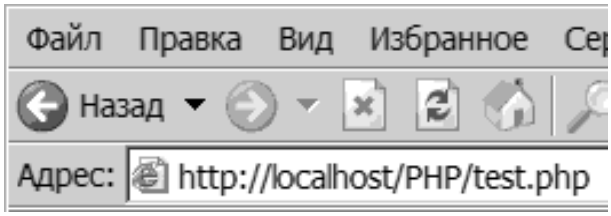

Рисунок 4.17 – Результат виконання прикладу для оператору continue

Зауважимо – якщо умовний оператор знаходиться після операторів echo, код буде помилковим, і виведуться всі цифри від 1 до 20, оскільки перевірка умови виходу з циклу на даній ітерації відбуватиметься вже після виконання цієї ітерації.

Нескінченний цикл реалізується за допомогою оператора while таким чином:

```
while(1)
  {
     ...
  }
```

Це теж саме, що і запис while(true).

Оператори циклу Do…while. Цей оператор називається оператором циклу з постумовою. При вході в цикл у будь-якому випадку виконується тіло циклу (тобто цикл завжди буде виконаний хоча б один раз), потім обчислюється умова, і якщо вона не рівна 0, знов виконується тіло циклу. У нижченаведеному прикладі нуль завжди буде доданий в список, незалежно від умови (++$i <= $var) (рисунок 4.18).

```
1   <?
2   $var = 5;
3   $i = 0;
4   do
5   {
6   echo($i); echo('<br>');
7   }
8   while(++$i <= $var)
9   ?>
```

Рисунок 4.18 – Приклад застосовування оператора циклу do…while

Результат прикладу застосовування оператора циклу do…while (рисунок 4.19):

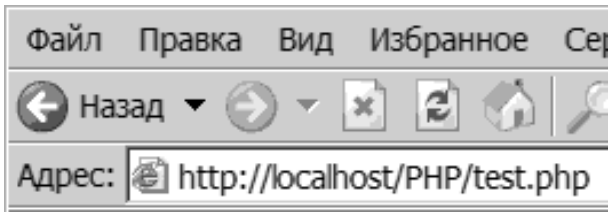

Рисунок 4.19 – Результат виконання прикладу для оператору do…while

Цикл з післяумовою буває корисний при обробці деяких послідовностей, коли обробку потрібно закінчувати не до, а після появи кінцевої ознаки.

Нескінченний цикл реалізується так:
do; while(1);

Оператори циклу for. Ітераційний цикл має наступний формат:

```
for(expression1;expression2;expression3)
{
        statements;
}
```

Тут expression1 (ініціалізація циклу) – послідовність визначень і виразів, що розділяється комами. Всі вирази, що входять в ініціалізацію, обчислюються тільки один раз при вході в цикл. Як правило, тут встановлюються початкові значення лічильників і параметрів циклу. Значення виразу-умови (expression2) такої ж як і у циклів з перед- і постумовами. За відсутності виразу-умови передбачається, що його значення завжди істинне. Вирази expression3 обчислюються в кінці кожної ітерації після виконання тіла циклу.

У наступному скрипті, ми за традицією виведемо числа від 0 до 5 (рисунок 4.20):

```
1  <?
2  $var = 5;
3  $i = 0;
4  for ($i = 0; $i <= $var; $i++)
5  {
6  echo($i);
7  echo('<br>');
8  }
9  ?>
```

Рисунок 4.20 – Приклад застосовування оператора циклу for

Результат аналогічний, показаному на попередньому малюнку.
Нескінченний цикл можна організувати таким чином:

```
for(;;);
```
або
```
for(;1;);
```

Функція isset()
```
isset(variable);
```

Функція isset() – корисне доповнення до оператору if, яка дозволяє визначити, чи встановлено значення змінної. Наприклад, перевірка введення

користувачем інформації у формі (рисунок 4.21). Також можна написати простий оператор if для контролю за тим, чи натиснув користувач кнопку Submit. Якщо змінна $Submit була встановлена, то можна приступити до обробки даних з форми. Якщо ж змінна $Submit не була встановлена, це означає, що користувач ще не предоставив інформацію у формі і ви повинні показати цю форму в браузері, що відвідувач вузла міг ввести дані.

```
1   if (isset ($submit)):
2   //Виконати якісь дії
3   else:
4   //Вивести форму
5   endif;
```

Рисунок 4.21 – Перевірка введення користувачем інформації у формі

З функцією isset() можливо також використовувати оператор «НІ» (!) для перевірки того, чи заповнені всі поля форми (рисунок 4.22).

```
1   if (isset ($phone_nambe)):
2   echo “Ви не ввели свій номер телефону!\n”;
3   endif;
```

Рисунок 4.22 – Приклад застосовуння оператора «НІ»

Створення користувацьких функцій. Коли ми здійснюємо дії, в яких простежується залежність від якихось даних, і при цьому, можливо, нам знадобиться виконувати такі ж дії, але з іншими початковими даними, зручно використовувати механізм функцій – оормити блок дій у вигляді тіла функції, а змінні дані - як її параметри.

Щоб створити свою функцію достатньо надати ім’я функції, перерахувати параметри, які їй передаються та записати оператори, з яких вона буде створена. Зазвичай параметрами функції є дані, які їй треба обробити. Також потрібно оператори функції взяти у фігурні дужки.

```
Function name (arg1, arg2, arg3, …)
{
Оператори;
}
```

Приклад об’явлення функції:

```
Function name_print ($name)
{
Echo “<p>Ім’я:<b> $name</b>”;
}
```

Їй передається один параметр – змінна $name, значення якої і виводиться в браузер.

Щоб викликати цю функцію, достатньо вказати її ім’я та передати ту ж кількість параметрів, яку була означено при її створенні. Вдпнному прикладі передається тільки один параметр.

Значимість користувацьких функцій у тому, що економиться час при створенні сценаріїв. Замість того, щоб кожний раз, коли треба ввести ім’я у початковому тексті замість оператора

Echo “<p>Ім’я:$name”;

достатньо ввести name_print ($name);

Хід роботи

1. Відтворити приклади, що наведені у лабораторній роботі.
2. Розробити сценарій, який буде перевіряти чи заповненні обов’язкові поля форми. Форма буде збирати інформацію про користувача, а саме: ім’я, прізвище, адресу електронної скриньки, поштовий код та назву улюбленого предмету (рисунок 4.23).

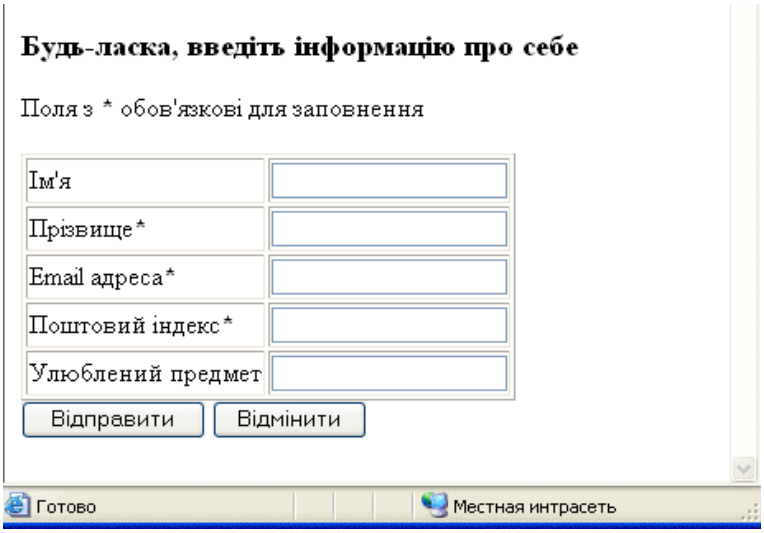

Рисунок 4.23 – Зовнішній вигляд форми для завдання 2

2.1. Створити три функції. Перша - print_form просто виводить форму на екран. Якщо деякі поля вже були заповнені, функція автоматично вставляє в них належні значення. Це зручно, так як користувачу не потрібно знову і знову вводити одну й ту ж саму інформацію тільки через те, що він забув заповнити одне з обов’язкових полів.

2.2. Друга функція - check_form перевіряє заповнення обов’язкових полів. Якщо якесь з них не заповнено, функція повідомляє про це користувачу, а потім викликає print_form для повторного відображення форми (рисунок 4.24).

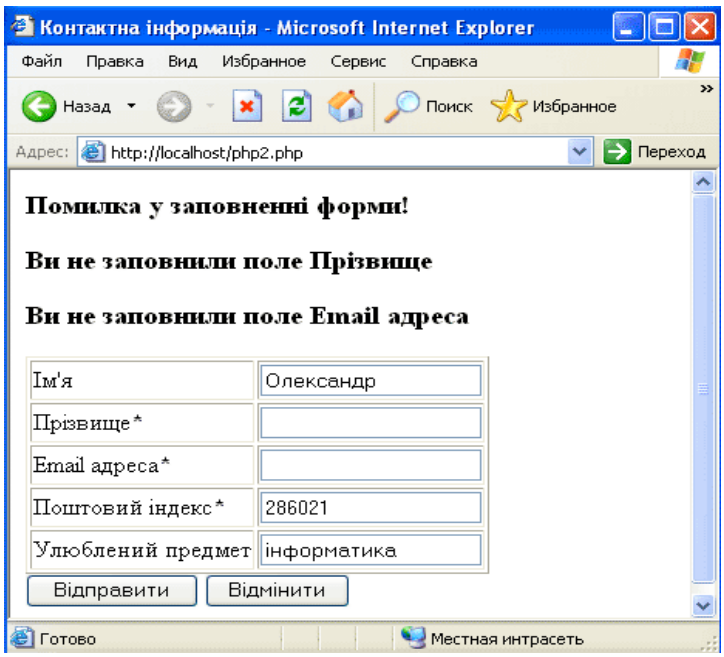

Рисунок 4.24 – Результат роботи сценарію, якщо у формі були знайдені помилки

2.3. Третя функція confirm_form, просто відображає введену користувачем інформацію (рисунок 4.25).

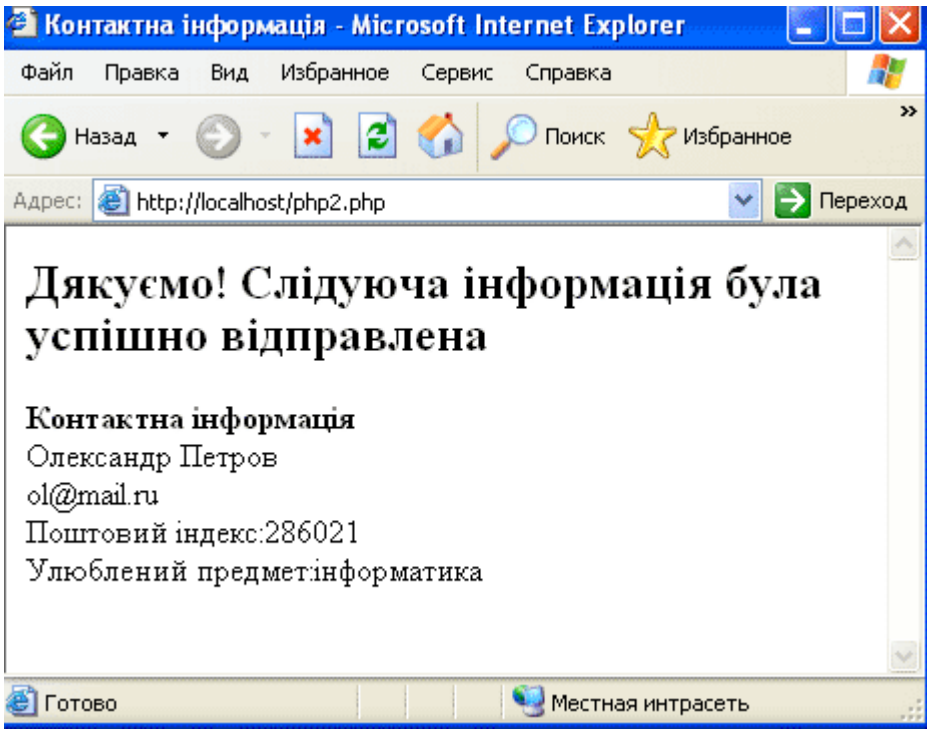

Рисунок 4.25 – Результат роботи сценарію, якщо інформація представлена правильно

Текст самого сценарію наведено нижче (рисунок 4.26):

```
1   <html>
2   <head>
3   <title>Контактна інформація</title>
4   </head>
5   <body>
6   <?php
7   /* Об'являємо деякі функції*/
8   function print_form ($f_name, $l_name, $email, $zip, $object){
9   ?>
10  <form  action="form_checker.php" method="post">
11  <table cellspasing="2" cellpedding="2" border="1">
12  <tr>
13  <td>Ім'я</td><td><input name="f_name" type="text"
14  value="<?php echo $f_name ?>"></td>
15  </tr>
16  <tr>
17  <td>Прізвище<b>*</b></td><td><input name="l_name"
18  type="text" value="<?php print $l_name ?>"></td>
19  </tr>
20  <tr>
21  <td>Email адреса<b>*</b></td><td><input name="email" type="text" value="<?php
22  print $email ?>">
23  </td>
24  </tr>
25  <tr>
26  <td>Поштовий індекс<b>*</b></td>
27  <td><input name="zip"
28  type="text" value="
29  <?php print  $zip ?>">
30  </td>
31  </tr>
32  <tr>
33  <td>Улюблений предмет</td>
34  <td><input name="object" type="text" value="
35  <?php print  $object ?>">
36  </td>
37  </tr>
38  </table>
39  <input name="submit" type="submit" value="Надіслати">
40  <input type="reset" value="Відмінити">
41  </form>
42  <?
43  }
44  function check_form ($f_name, $l_name, $email, $zip, $object){
45  if (!$l_name||!$email||!$zip):echo "<h3>Помилка у заповненні форми!</h3>";
46  if (!$l_name){
```

Рисунок 4.26 – Текст самого сценарію, аркуш – 1

```
47  echo "<h3>Ви не заповнили поле <b>Прізвище</b></h3>";
48  }
49  if (!$email){
50  echo "<h3>Ви не заповнили поле <b>Email адреса</b></h3>";
51  }
52  if (!$zip){
53  echo "<h3>Ви не заповнили поле <b>Поштовий індекс</b></h3>";
54  }
55  print_form ($f_name, $l_name, $email, $zip, $object);
56  else:
57  confirm_form ($f_name, $l_name, $email, $zip, $object);
58  endif;
59  }
60  function confirm_form ($f_name, $l_name, $email, $zip, $object){
61  ?>
62  <h2>Дякуємо! Слідуюча інформація була успішно надіслана      </h2>
63  <b>Контактна інформація</b>
64  <?
65  echo "<br>$f_name $l_name<br>$email<br>Поштовий  ідекс:$zip<br>
66  Улюблений предмет:$object\n";
67  }
68  /* Початок основної програми*/
69  if (!$submit):
70  ?>
71  <h3>Будь-ласка, введіть інформацію про себе</h3>
72  Поля з <b>*</b> обов'язкові для заповнення<p>
73  <?php
74  print_form("","","","","","");
75  else:
76  check_form($f_name, $l_name, $email, $zip, $object);
77  endif;
78  ?>
79  </body>
80  </html>
```

Рисунок 4.26 – Текст самого сценарію, аркуш - 2

Вміст звіту

а) тема і мета лабораторної роботи;
б) у відповідності до кроків ходу роботи навести:
   1) графічне представлення інженерних дій у вигляді схем програм, схем взаємодії програмних модулів, діаграм використання або інших UML-діаграм;
   2) лістинг всіх відлагоджених програмних модулів;
   3) результати виконання програмних модулів;
в) змістовні висновки за результатами роботи.

Питання для самоконтролю

1. Які оператори умовного переходу ви знаєте?
2. Як працює оператор (if...else)?
3. Для чого використовують elseif?
4. Яка операція застосовується для об'єднання двох умов в одне?
5. Як можна замінювати блоки if…else умовною операцією?
6. Для чого використовують перемикач (switch)?
7. Які оператори циклу ви знаєте?
8. Як працює цикл з передумовою?
9. Як працює цикл з постумовою?
10. Як працює ітераційний цикл?
11. Як працює функція isset?
12. Що таке користувацька функція? Наведіть приклад.
13. Як об’явити користувацька функцію?
14. Які користувацькі функції ви використовуєте в 2 завданні?
15. Для чого створена функція function print_form?
16. Для чого створена функція function check_form?
17. Для чого створена функція function confirm_form?

ЛАБОРАТОРНА РОБОТА №5. ДОСЛІДЖЕННЯ ОПЕРАТОРІВ ПРИЄДНАННЯ ТА АНАЛІЗУ БАЗИ ДАНИХ В ТЕХНОЛОГІЇ PHP-MySQL

Мета: набуття навичок створення простих серверних скриптів та аналізу баз даних і таблиць.

Теоретичні відомості

Для проведення досліджень будуть використані наступні SQL-функції, а також скриптова мова програмування PHP. Функції наведені в синтаксисі мови програмування PHP.

Функція mysql_connect() Функція mysql_connect - Відкриває з'єднання з сервером MYSQL.

Повертає покажчик на з'єднання з MYSQL у разі успішного виконання, або FALSE при невдачі.

При її виклику, в якості параметра повинні бути вказані місцезнаходження сервера, реєстраційне ім'я і пароль користувача MYSQL. Значення, яке повертає mysql_connect(), зазвичай привласнюється змінній, яка потім використовується функцією mysql_select_db():

$server_connection = mysql_connect (“localhost”, “username”, “password”);

З'єднання з сервером буде закрито при завершенні виконання скрипта, якщо до цього воно не буде закрито за допомогою функції mysql_close().

Функція mysql_select_db() - вибирає базу даних MYSQL:

mysql_select_db(“news”, “$server_connection”);

Повертає TRUE у разі успішного завершення, FALSE у разі виникнення помилки. Mysql_select_db() вибирає для роботи вказану базу даних на сервері, на який посилається переданий покажчик. Якщо параметр покажчика опущений, використовується останнє відкрите з'єднання. Якщо немає жодного відкритого з'єднання, функція спробує з'єднатися з сервером аналогічно функції mysql_connect(), викликаною без параметрів.

Функція mysql_query() - посилає запит MYSQL.

Mysql_query() надсилає запит до активної бази даних сервера.

Тільки для запитів SELECT, SHOW, EXPLAIN, DESCRIBE, mysql_query() повертає покажчик на результат запиту, або FALSE якщо запит не був виконаний. У решті випадків, mysql_query() повертає TRUE у

разі успішного запиту і FALSE у разі помилки. Значення не рівне FALSE говорить про те, що запит був виконаний успішно.

$result = mysql_query (“SELECT * FROM news”,$db);

Функція mysql_fetch_array приймає перший рядок результату виконання MYSQL і поміщає його в масив. При кожному виклику цієї функції повертається наступний рядок даних з результатів запиту до тих пір, поки не будуть оброблені всі рядки. На масив можна посилатися по номеру або по імені стовпця:

$myrow = mysql_fetch_array($result);

Запит SHOW DATABASES призначений для одержання інформації про доступні користувачеві бази даних:

$result = mysql_query("SHOW DATABASES",$db);

Для того, щоб почати роботу з таблицями, необхідно повідомити MySQL з якою базою даних ви маєте намір працювати. Це здійснюється за допомогою команди USE:

mysql_select_db('my_db', $db);

Тут my_db - назва вибираної бази даних.

Запит DESCRIBE дозволяє одержати інформацію про структуру вказаної таблиці і має наступний синтаксис:

$x = mysql_query("DESCRIBE table_name",$db);

Тут table_name - назва таблиці, структуру якої треба одержати. Наприклад, може бути одержана структура такої таблиці :

Таблиця 5.1 – Структура таблиці

Field	Type	Null	Key	Default	Extra
news_id	int(11)		PRI	0	auto_increment
heading	varchar(48)	YES		NULL	
body	text	YES		NULL	
date	date	YES		NULL	
author_name	varchar(48)	YES		NULL	
author_email	varchar(48)	YES		NULL	

Короткий опис елементів таблиці:

- кожна стрічка в графі Field повідомляє про назву поля (стовпця таблиці), в якому містяться дані;
- в графі Type розміщена інформація про те, які дані і якого типа можуть зберігатися в кожному конкретному полі таблиці;
- якщо в графі NULL відповідного поля стоїть слово YES, то значення цього поля може бути рівним NULL;
- наявність напроти конкретного поля в графі Key позначення PRI (скорочення від слова primary – первинний ключ) вказує на те, що дане поле є індексним. У стовпці первинного ключа не можуть знаходитися два однакових значення;
- графа Default вказує на те, які значення вводяться в це поле за замовчуванням;
- і, нарешті Extra, описує будь-які додаткові атрибути, які може мати відповідна графа таблиці. Наприклад, атрибут auto_increment, призначений первинному ключу, вказує, що значення збільшується на 1 кожного разу при додаванні нового рядка до таблиці.

Функція include(filename). Функція include() виконує саме те, про що повідомляє її назва: тобто включає в текст скрипту інший файл. Вона вельми зручна, оскільки дозволяє використовувати заздалегідь підготовлені фрагменти коду, які можуть бути вставлені в різні сторінки, але зберігаються в одному місці. Синтаксис функції:

include(“file.inc”);

Якщо ви помістите вище приведений рядок до одного з своїх скриптів, то зможете застосувати будь-які функції, визначені у файлі file.inc. В дійсності можна використовувати не тільки функції, але й будь-яку інформацію, що міститься у вказаному файлі. Ви можете вставити файл майже будь-якого типу, наприклад, що містить текст, код HTML або код PHP. Файли, що включаються в сценарій, як правило, мають розширення .inc, що спрощує їх пошук.

Завдання

1. Користуючись власним ftp-обліковим записом, створити на сервері PHP-скрипт згідно інструкцій, що наведені нижче, для дослідження функцій MySQL.
2. Провести дослідження основних функцій створення, перегляду та зміни баз даних та структури таблиць з використанням LAMP (Linux-Apache-Mysql-PHP) технології.

Хід роботи

1. Завантажити операційну систему Linux.
2. Запустити файловий менеджер (ФМ) Krusader і приєднатись за протоколом ftp до сервера, використовуючи логін і пароль свого власного ftp-облікового запису. В з'єднанні використати сервер 192.168.1.5 або vsau.vin.ua (Нагадаємо, що для вмикання ftp-з'єднання в ФМ Krusader`і треба або натиснути Ctrl_N або ж скористатись його меню).
3. Після успішного з'єднання із сервером, користуючись меню ФМ, створіть новий файл на ftp-панелі з назвою lr5.php. Одразу після цього має відкритись текстовий редактор. Збережіть цей порожній текст (Ctrl_S) і далі внесіть до нього наступний PHP-код (рисунок 5.1).

В цьому скрипті після з'єднання із сервером в змінній $db буде знаходитись код з'єднання. За одержаним кодом ми надалі будемо спілкуватись із MySQL-сервером

Функція mysql_query призначення для надсилання команд MySQL-серверу. Після її виконання маємо результат, який надається змінній (наприклад $x). У більшості випадків результатом є масив стрічок для того що би їх виділити використовується функція mysql_fetch_row, яка ... також дає масив стрічок. Але їх вже можна використовувати для роботи.

На операторі while створений цикл, який повторює дію стільки разів, скільки в масиві (в нашому прикладі це $a) буде стрічок. Зверніть увагу, що в його круглих дужках while($a = mysql_fetch_row($x)) не тільки працює функція mysql_fetch_row, але й змінні $a надається значення і на останок оператор while перевіряє, чи не порожній масив повернула функція mysql_fetch_row! До речі, в процесі кожного while-повторення запускається функція mysql_fetch_row, яка отримує наступний елемент масиву $x.

До кожного одержаного елементу стрічкового масиву треба звертатись за індексом, наприклад $a[1] або $a[0].

Функція die використовується для формування повідомлень, якщо з тих чи інших причин не спрацює оператор запиту до MySQL-серверу.

4. Після успішного набору скрипта і його збереження, перевірте його роботу web-браузером. Для цього відкрийте такий url : http://81.30.162.20/lamp/XXX/lr1.php, де XXX це назва вашого ftp-облікового запису, наприклад, ek1. В разі відсутності помилок в скрипті, ви побачите WEB-сторінку, яку згенерував скрипт.

```
1   <?php
2   include("../inc.php");
3    $db = mysql_connect($cServname, $cUsername, $cPassword) or die ('Не
4    можу приєднатись до MySQL-серверу.');
5   print "Переглянемо доступні нам бази даних. (SHOW DATABASES)<hr>";
6   $x = mysql_query("SHOW DATABASES",$db) or die (mysql_error());
7   while($a = mysql_fetch_row($x)) {
8   print "$a[0]<br>";
9   }
10  // приєднуємось до бази даних $cDatabase (у вашому випадку це БД "student")
11  mysql_select_db( $cDatabase, $db ) or die ("Не можу активізувати базу даних");
12  // і з'ясуємо, що там є ...
13  print "<br>Переглянемо список наявних таблиць в базі даних $cDatabase. (Show
14  Tables)<hr>";
15  $x = mysql_query("SHOW TABLES",$db) or die (mysql_error());
16  while($a = mysql_fetch_row($x)) {
17  print "$a[0]<br>";
18  }
19  // а тепер подивимось, яку структуру має таблиця
20  $t='news'; // надамо змінній $t назву однієї з наявних в базі баних таблиць
21  print "<br>Переглянемо структуру таблиці \"$t\" (DESCRIBE TABLE)<hr>";
22  print "<table border='1'>
23  <th>Назва поля</th>
24  <th>Тип поля</th>
25  <th>Нуль</th>
26  <th>Первинний ключ</th>
27  <th></th>
28  <th>Спосіб зміни ключа таблиці</th>";
29  $x = mysql_query("DESCRIBE $t",$db) or die (mysql_error());
30  while($a = mysql_fetch_row($x)) {
31   print "<tr>
32   <td> $a[0]</td>
33   <td> $a[1]</td>
34   <td> $a[2]</td>
35   <td> $a[3]</td>
36   <td> $a[4]</td>
37   <td> $a[5]</tr>";
38   }
39   print "</table>";
40   // закриємо з'єднання із сервером
41   mysql_close($db);
42   ?>
```

Рисунок 5.1 – Вміст PHP-коду

Вміст звіту

а) тема і мета лабораторної роботи;
б) у відповідності до кроків ходу роботи навести:

1) графічне представлення інженерних дій у вигляді схем програм, схем взаємодії програмних модулів, діаграм використання або інших UML-діаграм;
2) лістинг всіх відлагоджених програмних модулів;
3) результати виконання програмних модулів;

в) змістовні висновки за результатами роботи.

Питання для самоконтролю

1. Для чого існує функція include(). Де і в яких сценаріях вона використовується?
2. Яка інформація міститься у файлі inc.php?
3. Яка інформація міститься у файлі inc_tab.php?
4. Яке призначення функції mysql_connect()?
5. Яке призначення функції mysql_select_db()?
6. Яке призначення функції mysql_select_db()?
7. Яке призначення функції mysql_query()?
8. Яке призначення функції mysql_fetch_array ()?
9. Як працює цикл в операторі while(...) і які при цьому виконуються функції?
10. Що таке асоціативний масив? Де в скрипті вони використовуються?
11. Що може виступати у якості індексу асоціативного масиву?
12. Як працює та частина скрипту, де створюється таблиця з призначеннями полів? Які HTML-теги створюють таблицю, її заголовок, рядки, колонки?
13. Чому в скрипті, де виводиться таблиця, ніде не вказана кількість її рядків? Чим вона тоді визначається?

ЛАБОРАТОРНА РОБОТА №6. СТВОРЕННЯ ТА ЗАПОВНЕННЯ РЕЛЯЦІЙНОЇ БАЗИ ДАНИХ В ТЕХНОЛОГІЇ PHP-MySQL

Мета: набуття навичок створення та редагування баз даних і таблиць.

Теоретичні відомості

Як відомо, бази даних складаються як правило з таблиць. Кожна з них має свою назву і структуру, тобто певний набір полів і їх назв. Кожне поле, як і змінні в програмуванні має свій тип. Типізація полів призначена для прискорення обробки інформації сервером. Адже знаючи тип даних в полі, наприклад цілі числа, можна одразу застосовувати до них цілочисельну математику, не витрачаючи час на визначення типу даних.

Список типів полів, що найчастіше зустрічаються, наведений в таблицях 6.1 – 6.3. Для багатьох з них надається максимальна розрядність або ширина поля, що вказується в дужках, яку ми далі позначатимемо символом як max. Наприклад, запис INT(2) означає, що значення даного поля не може перевищувати 99.

До числових типів відносяться цілі числа і дійсні числа (числа з плаваючою крапкою). Для чисел з плаваючою крапкою, окрім максимальної ширини розрядності або ширини поля можна також вказувати число значущих цифр після коми, що в таблиці 6.1 позначається символом P.

Таблиця 6.1 – Числові типи

Тип	Опис
TINYINT[(max)]	Дуже маленькі цілі числа діапазону -127…128.
SMALLINT[(max)]	Малі цілі числа діапазону -32768…32767.
MEDIUMINT[(max)]	Середні цілі числа.
INT[(max)]	Звичайні цілі числа.
FLOAT[(max,P)]	Числа з плаваючою точкою.
DOUBLE[(max,P)]	Числа з плаваючою точкою подвійної точності.
DECIMAL[(max,P)]	Числа з плаваючою крапкою, що приведені до типу char.

Рядкові типи полів наведені в таблиці 6.2.

Таблиця 6.2 – Рядкові типи

Тип	Опис
CHAR (len)[BINARY]	Рядки з довжиною len, яка не перевищує 255 символів. Ключове слово BINARY вказує на те, що дані повинні оброблятися незалежно від регістра.
CHAR	Синонім CHAR(1).
VARCHAR (len)[BINARY]	Синонім CHAR(len) за винятком того, що рядки можуть бути довільної довжини.
TEXT	Рядки з максимальною довжиною – 65535 байтів. Дані цього типу чутливі до регістру.
BLOB	Бінарні (двійкові) дані з об'ємом до 65535 байтів. Тип BLOB (binary large object - великий двійковий об'єкт) призначений для зберігання будь-яких двійкових даних, зокрема зображень і звукових послідовностей.

Різні типи для зберігання дати і часу приведені в таблиці 6.3.

Таблиця 6.3 – Типи дати і часу

Тип	Опис
DATE	Дата у форматі РРРР-ММ-ДД.
TIME	Час у форматі ЧЧ-ММ-СС.
DATETIME	Дата і час у форматі РРРР-ММ-ДД ЧЧ-ММ-СС.
YEAR	Рік у форматі РР або РРРР.
TIMESTAMP	Мітка часу для відліків за транзакціями у форматі РРРР-ММ-ДД ЧЧ-ММ-СС.

Створення таблиць (команди нижче надаються в синтаксисі PHP):

1. CREATE TABLE table_name – створити таблицю із заданою назвою (тут table_name – назва створюваної таблиці). В цій команді також треба вказати структуру майбутньої таблиці, тобто назви полів та їх типи. Додатково можуть бути вказані ключові поля, тип кодування символів, а також коментарі. Наприклад, створимо таблицю news з полями news_id, heading, body, date, author_name, author_emaile і ключовим полем (PRIMARY KEY) news_id (рисунок 6.1).

```
1  mysql_query("CREATE TABLE news (
2  news_id INT NOT NULL auto_increment,
3  heading VARCHAR(48),
4  body TEXT,
5  date DATE,
6  author_name VARCHAR(48),
7  author_email VARCHAR(48),
8  PRIMARY KEY(news_id))", $db);
```

Рисунок 6.1 – Створення таблиці за допомогою команди CREATE TABLE

Для зручності запису і читання цей рядок з командою записаний у декілька стрічок.

Первинному ключу (PRIMARY KEY) можна наданий атрибут auto_increment, який буде в майбутньому *автоматично* генерувати унікальний ключ до кожного нового запису. Цей ключ представляє собою ціле неповторюване в інших записах цієї таблиці число. auto_increment – означає, що воно формується шляхом збільшення на одиницю значення останнього ключа. При вставці першого запису до таблиці значення ключа виставляється рівним нулю.

2. Зміна структури вже існуючих таблиць ALTER TABLE. Ця команда має три методи:

- додавання нового поля до таблиці (Add). Наприклад:
  mysql_query("ALTER TABLE birthdays ADD age CHAR(3)", $db);
  Тут до структури таблиці буде додане ще одне поле age типу CHAR довжиною 3 байта;
- змінити поле (CHANGE). Наприклад, встановимо в таблиці news довжину поля author_name у 15 байтів:
  mysql_query("ALTER TABLE news CHANGE author_name author_name VARCHAR(15)", $db);
  Або ж змінимо назву поля body в таблиці news на нову - description:
  mysql_query("ALTER TABLE news CHANGE body description");
- видалити поле (DROP). В прикладі показано видалення поля date в таблиці news.
  mysql_query("ALTER TABLE news DROP date", $db);
  Змінити назву таблиці (RENAME TABLE). Наприклад,
  mysql_query("RENAME TABLE news TO story ", $db);
  Видалити таблицю DROP TABLE
  mysql_query("DROP TABLE table_name", $db);

Видаляється таблиця із вказаною назвою, наприклад table_name. Будьте обережні! Відновлення видаленої таблиці неможливе!

Додавання нового запису (INSERT).

В цій команді треба вказати назву таблиці (наприклад news), до якої потрібно вставити величини, а також вказати значення ВСІХ (це обов'язково) полів цього запису. В разі, якщо значення немає, то треба передати порожнє значення, наприклад " . Значення полів передаються оператором VALUES. В його круглих дужках через кому перераховуються значення цього запису. Кількість значень в операторі VALUES обв'язково має збігатись із кількістю полів в таблиці. Вставимо, наприклад порожній запис в таблюцю news. Там 6 полів. Тоді маємо виконати :

mysql_query("INSERT INTO news VALUES(","","","","","), $db ");

Цікаво ось що. Після додавання нового запису в першому полі (насправді їх нумерація йде з 0) буде не порожній рядок, а число 0! Чому? Вище, коли ми створювали структуру таблиці news це поле було вказано як news_id INT NOT NULL auto_increment. Сам auto_increment при додаванні нових записів до цього поля буде самостійно буде вносити ключове число.

А тепер додамо до цієї ж таблиці ще один запис з даними (для зручності читання запишемо оператор декількома рядками)(рисунок 6.2).

```
1  mysql_query("INSERT INTO news VALUES(
2  '',
3  'Назва',
4  'Текст новин',
5  '11-16-2006',
6  'Загоруйко В.',
7  'sergiy@ukr.net'
8  ) , $db ");
```

Рисунок 6.2 – Приклад додавання до таблиці ще одного запису з даними

Замість конкретних значень, можна підставляти і змінн (рисунок 6.3).

```
1  mysql_query("INSERT INTO news VALUES(
2  '',
3  '$a',
4  'Текст новин',
5  '$dt',
6  'Загоруйко В.',
7  '$qwerty'
8  ) , $db ");
9  Зміна запис-у(ів) (UPDATE)
```

Рисунок 6.3 – Приклад додавання до таблиці запису з конкретними значеннями

Зрозуміло, що, якщо уявити собі таблицю із записами, представити, що в ній треба щось змінити, то виникають такі питання:

1. В якому записі зробити зміни?
2. В якому полі зробити зміни?
3. Які дані треба занести?

Команда UPDATE дозволяє змінювати значення в таблиці. Для цього їй треба вказати назву таблиці, назву поля(ів) і значення, яке треба в ньому (у них) замінити. Наприклад, (рисунок 6.4):

```
1    mysql_query("UPDATE news SET
2    body = 'В Україні настала гарна погода ',
3    author_name = 'Василенко М.А.' ",
4    $db);
```

Рисунок 6.4 – Приклад застосування команди UPDATE

Із наведеного зрозуміло, що до вказаних полів (body і author_name) маю внестисть відповідні значення. Але ж дещо ми забули! Тут не вказано, в ЯКОМУ записі треба зробити ці зміни? Помилка? Ні. Просто ці значення внесуться до ВСІХ записів, що є у таблиці.

Як вказати серверу, ДЕ треба робити зміни. Для цього існує оператор умови WHERE (перекладається “в тому випадку коли”). В мові PHP на ньому створюються цикли. А в мові запитів від використовується для вказування умови (рисунок 6.5).

```
1    mysql_query("UPDATE news SET
2    body = 'Зібраний гарний урожай ',
3    author_name = 'Хлопецька О.Ф.'
4    WHERE news_id='1' ",
5    $db);
```

Рисунок 6.5 – Приклад застосовуння оператора умови WHERE

Тепер зміни будуть внесені лише до запису, який має в полі news_id значеня 1.

Кількість одночасно змінюваних полів і записів може бути будь-яка. А сама умова може бути і складною, як в операторі IF, наприклад:

```
WHERE author_email ='my@asd.com' AND heading='Рослини'
```

Відбір записів SELECT. SELECT – мабуть, найголовніший оператор, який використовується при роботі з базами даних (рисунок 6.6). Головна його функція – це вибірка даних з таблиць за вказаними умовами, наприклад:

$x=mysql_query("SELECT field_name FROM table_name", $db);

Запис означає “відібрати з таблиці table_name всі дані, що знаходяться в полі field_name”. Чому всі? Тому що не вказано умови відбору (оператор WHILE). Теж саме, але з умовою і відбір будемо робити за двома полями:

```
1   $x = mysql_query("SELECT news
2   body,
3   author_name
4   WHERE author_email ='my@asd.com' AND heading='Рослини' ",
5   $db);
```

Рисунок 6.6 – Приклад застосовування оператора умови SELECT

Звучить так: “З таблиці news відібрати інформацію з полів body і author_name з тих записів, де в полях author_email знаходиться стрічка 'my@asd.com' і в heading стрічка 'Рослини'.

А можна відбирати одразу з усіх полів і їх назви не вказувати? Звичайно, тільки замість назв треба розмістити символ * (рисунок 6.7).

```
1   $x = mysql_query("SELECT *
2   WHERE author_email ='my@asd.com' AND heading='Рослини' ",
3   $db);
```

Рисунок 6.7 – Приклад застосовування оператора умови SELECT з символом *

Залишилось лише вияснити, що ж потрапляє до змінної $x ? Відповідь буде така: “Двовимірний масив рядків” або просто таблиця. Тобто при відбиранні даних у вказану змінну потрапляє таблиця, в якій буде стільки полів, скільки було вказано у запиті (в перших двох випадках їх було 2 (body і author_name), а в останньому – 5). І ця таблиця буде мати стільки записів, скільки “відібрав” оператор SELECT.

Як же одержати кожне значення окремо? Виділення окремих значень із масиву запиту mysql_fetch_row.

Оператор mysql_fetch_row дозволяє виділяти окремі рядки(записи) із двовимірного масиву. Якщо один раз виконати його, наприклад $y=mysql_fetch_row($x)); , то до змінної $y потрапить перший рядок таблиці $x. Повторне виконання цього ж оператору призведе до того, що до змінної $y потрапить другий рядок таблиці $x.

Добре, а як же тоді одержати значення кожного елементу цього рядку, наприклад body? Дуже просто! З індексом цього елементу в масиві. Індекси нумеруються по порядку від 0 і далі. Тому, якщо ми зробимо запит, наприклад по 2-х полях, то таблиця $x , буде мати лише 2 поля з індексами 0

і 1. Тобто значення поля body будуть мати індекс 0. Виведемо перше знайдене на web-сторінку (рисунок 6.8):

```
1    $x = mysql_query("SELECT news
2    body,
3    author_name
4    WHERE author_email ='my@asd.com' AND heading='Рослини' ",
5    $db);
6    $y=mysql_fetch_row($x));
7    print “$y[0]”;
```

Рисунок 6.8 – Приклад застосування оператора mysql_fetch_row

А якщо відібралось декілька записів? Як визначити їх кількість? Один із способів полягає у створенні циклу з повторенням оператору mysql_fetch_row. Повторення буде продовжуватись до тих пір, поки масив $y не стане порожнім (рисунок 6.9):

```
1    while($myr=mysql_fetch_row($x))
2    {
3    print "$y[0] $y[1]<br> ";
4    }
```

Рисунок 6.9 – Приклад створення циклу з повторенням оператора mysql_fetch_row

Видалення записів DELETE. Команда DELETE просто видаляє запис (або записи), що відповідають вказаному критерію. Для видалення запису(ів) з таблиці треба вказати назву цієї таблиці і умову, які записи треба видалити, наприклад:

mysql_query("DELETE FROM news WHERE news_id > ’2’ , $db ");

У розглянутому випадку з таблиці видалений рядок, де значення news_id дорівнює 1. У таблиці можна також, наприклад, видалити рядки, в яких поле author_name містить значення «Загоруйко В.» і.т.д.

mysql_query("DELETE FROM news WHERE author_name = ’Загоруйко В’ , $db ");

Якщо не вказати ніякої умови, то в таблиці будуть видалені всі рядки:

mysql_query("DELETE FROM news , $db ");

Завдання

1. Користуючись власним ftp-обліковим записом, створити на сервері PHP-скрипт згідно інструкцій, що наведені нижче, для дослідження функцій MySQL.
2. Провести дослідження основних функцій створення, перегляду та зміни баз даних та структури таблиць з використанням LAMP (Linux-Apache-Mysql-PHP) технології.

Хід роботи

1. Завантажити операційну систему Linux.
2. Запустити файловий менеджер (ФМ) Krusader і приєднатись по ftp до сервера, використовуючи логін і пароль свого власного облікового запису. В з'єднанні використати сервер 192.168.1.5. (Нагадаємо, що для вмикання ftp-з'єднання в ФМ Krusader`і треба або натиснути Ctrl_N або ж скористатись його меню).
3. Після успішного з'єднання із сервером, користуючись меню ФМ, створіть новий файл на ftp-панелі з назвою lr6.php. Одразу після цього має відкритись текстовий редактор з цим файлом. Збережіть його (Ctrl_S) і внесіть до нього наступний PHP-код.

Для роботи з базою даних застосування PHP є менш трудомістким, гнучкішим і, що найважливіше, легше здійсненним за допомогою Web-браузера.

У основі взаємодії PHP та бази даних покладено прості принципи:

- спочатку необхідно під’єднатися до серверу БД та зареєструватись;
- потім слід вибрати базу даних, яка стане використовуватися;
- і, нарешті, - відправляти запити SQL на сервер для додавання, видалення і зміни даних.

Вам навіть не треба пам'ятати про те, що в кінці роботи потрібно закрити з'єднання з сервером. PHP виконає це за вас.

Зараз ви повинні побудувати таблицю news за допомогою PHP.

Створюємо файл inc_tab.php, який теж приєднується до скрипту, за допомогою функції include(рисунок 6.10):

```
1   <?
2   $news_table = 'news';
3   ?>
```

Рисунок 6.10 – Вміст файлу inc_tab.php

Ви бачите, що тут створюється змінна $news_table, яка запам’ятовує назву таблиці. Кожен студент повинен дати свою індивідуальну назву, яка б більше не повторювалась. Наприклад:

$news_table = 'ek21news';

Для зручності роботи в майбутньому створіть ще один скрипт header.php із тегами заголовку сторінок, які будуть генерувати наші скрипти (рисунок 6.11).

```
1   <html>
2   <head>
3   <meta http-equiv="Content-Type" content="text/html; charset=windows-1251">
4   </head>
```

Рисунок 6.11 – Вміст файлу header.php

Створюємо скрипт news.php. Наша таблиця Новин повинна мати такі поля:

- порядковий номер - news_id;
- заголовок новини – heading;
- зміст новини – body;
- дату створення – date;
- ім’я автора - author_name;
- email автора - author_ email.

Типи цих полів, їх розмір та сама структура таблиці була розглянута раніше. Також ви тут бачите застосування функцій, розглянутих на початку лабораторної роботи.

```
1   <?php
2   include('../inc.php'); //підключаємо файл з інформацією про підключення
3   include('inc_tab.php'); // підключаємо файл з інформацією про таблиці
4   include('header.php'); // підключаємо файл з заголовочною частиною web-сторінки
5   $db = mysql_connect($cServname,$cUsername,$cPassword) // під'єднатись до MySql
6   сервера
7   or die('Unable to connect to MySQL server.'); // або вивести повідомлення про
8   неможливість під'єднання
9   mysql_select_db( $cDatabase, $db ) or die ("Could not activate the database");
10  // створюємо таблицю News, яка збирає замітки новин і має такі поля
11  echo"<h4>Створюємо таблицю \"$news_table\" ...</h4>";
12  mysql_query("CREATE TABLE $news_table (
13  news_id INT NOT NULL auto_increment,
14  heading VARCHAR(48),
15  body TEXT,
16  dt VARCHAR(10),
```

Рисунок 6.12 – Вміст скрипту news.php, аркуш - 1

```
17  author_name VARCHAR(48),
18  author_email VARCHAR(48),
19  PRIMARY KEY(news_id))",
20  $db) or die(mysql_error()); //якщо запит не виконано - виводимо зміст помилки
21  echo"<h4>Таблиці успішно створені</h4>";
22  mysql_close($db);
23  ?>
```

Рисунок 6.12 – Вміст скрипту news.php, аркуш – 2

Створюємо скрипт database.php, який буде заносити новини у БД. Цей скрипт демонструє, наскільки легко розмістити інформацію в БД, використовуючи PHP. Достатньо ввести лише декілька коротких стрічок. З тридцяти стрічок коду в цьому скрипті, лише біля десяти відносяться до PHP. У інших звичайна розмітка HTML. Зразки форм, які з'являються при виконанні скрипту, показані на рисунку 6.13 та 6.14.

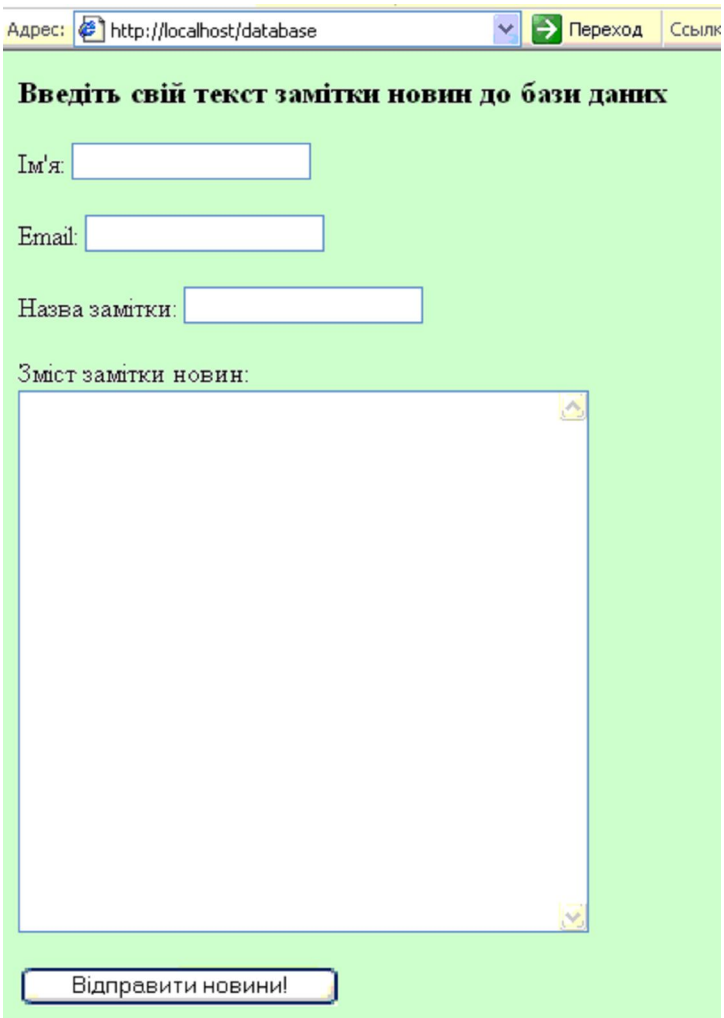

Рисунок 6.13 – Початкова форма введення інформації про новини в таблицю news з скрипту database.php

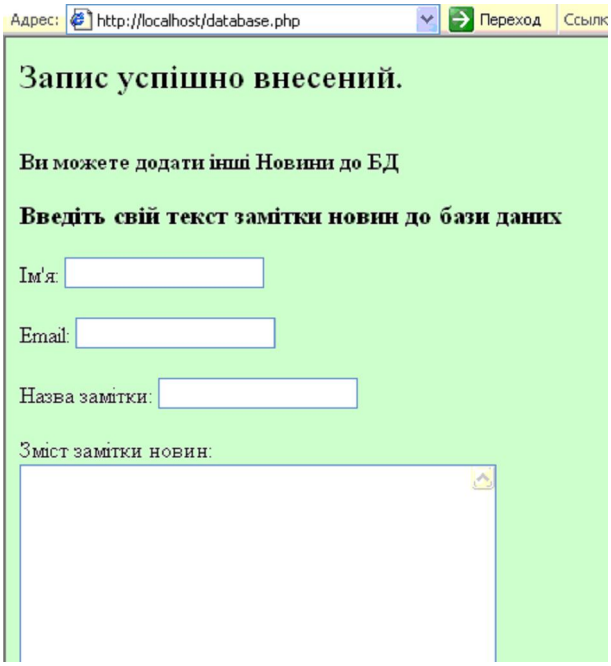

Рисунок 6.14 – Результат надсилання до сервера даних з форми при виконанні скрипту database.php

Текст самого скрипту database.php, зображено на рисунку 6.15.

В стрічці 2 під'єднуємо файл inc.php, в якому налаштована вся інформація для з'єднання з сервером БД.

В стрічці 3 під'єднуємо файл inc_tab.php, в якому зберігається назва таблиці.

В стрічці 4 під'єднуємо файл header.php, в якому зберігається назва таблиці.

В стрічці 7 перевіряється чи була натиснута кнопка Submit. Якщо так, то так, то виконаються рядки у фігурних дужках. Тобто додати до своєї таблиці новий запис.

В стрічці 8 результат виконання функції mysql_connect(...) надається змінній $db. Надалі значення цієї змінної ("канал зв'язку") використовується для діалогу скрипту із сервером баз даних MySQL.

В стрічці 10 функція mysql_select_db обирає для подальшої роботи базу даних, назва якої знаходиться у змінній $cDatabase.

У стрічках 13-25 виконується запит, згідно якого до таблиці має бути доданий новий запис із величинами в полях, що визначаються змінними $heading, $body, $date, $auth, $auth_email.

Починаючи із стрічки 28 за допомогою коду HTML створюється форма для введення інформації і надсилання її цьому ж самому скриптові.

Цей скрипт викликає сам себе (стрічка 35). Тому в одному сеансі може бути введено декілька заміток з новинами. Форма HTML виводиться на екран при кожному виконанні скрипту, та після чергового натискання користувачем кнопки Submit.

```
<?php
include('../inc.php');
include('inc_tab.php');
include('header.php');
print”<body bgcolor='#C4FFC4'>”
/* Ця програма дозволяє додавати Новини у базу даних */
if (isset ($submit)) { //якщо була натиснута кнопка submit на формі
$db = mysql_connect($cServname,$cUsername,$cPassword) or die (“Не можу
приєднатись до MySQL-сервера. Перевірте чи не порожні значення змінних.”);
mysql_select_db( $cDatabase, $db ) or die ("Не можу приєднатись до бази даних:
$cDatabase.");
$date=date("Y-m-d");
if ( mysql_query("INSERT INTO $news_table VALUES (
NULL,
'$heading',
'$body',
'$date',
'$auth',
'$auth_email'
)", $db) ) {
echo "<h2>Запис успішно внесений.</h2>";
echo "<br><b>Ви можете додати інші Новини до БД</b>";
} else {
echo "<br><b>Трапилась помилка. Перевірте скрипт.</b>";
}
}
?>
<p><h3>Введіть свій текст замітки новин до бази даних</h3></p>
<form action="database.php" method="post">
Ім'я: <input name="auth" type="text" ><p></p>
Email: <input name="auth_email" type="text"><p></p>
Назва замітки: <input name="heading" type="text"><p></p>
Зміст замітки новин: <br>
<textarea name="body" rows=20 cols=40></textarea><p></p>
<input type="submit" name="submit" value="Надіслати новини!">
</form>
</body>
</html>
```

Рисунок 6.15 – Вміст скрипту database.php

4. Самостійно додати засоби контролю помилок, описані у попередній роботі.

5. Додайте 8-10 записів до БД.

6. Для того, щоб переглянути зміст таблиці у БД, напишіть наступний скрипт view_news.php (рисунок 6.16).

```
1   <?php
2   include('../inc.php');
3   include('inc_tab.php');
4   include('header.php');
5   print”<body bgcolor='#C4FFC4'>”
6   /* Ця програма дозволяє переглядати зміст таблиці Новин у базі даних */
7   $db = mysql_connect($cServname,$cUsername,$cPassword) or die (“Не можу
8   приєднатись до MySQL-сервера. Перевірте чи не порожні значення змінних.”);
9   mysql_select_db( $cDatabase, $db ) or die ("Не можу приєднатись до бази
10  даних: $cDatabase.");
11  $date=date("Y-m-d");
12  echo "<h2>Таблиця $news_table $date</h2>";
13  /* Тут запам’ятовуємо у змінну $rezult запит до серверу */
14  $rezult= mysql_query("SELECT * FROM $news_table ");
15   /* Змінна $items приймає перший рядок результату виконання MYSQL і розміщує
16  його в масиві рядків. Щоб доступитись до елементів масиву використовуємо
17  цикл while */
18  while ($items = mysql_fetch_row($rezult))
19  /* Виводимо результат у вигляді таблиці на екран, де у кожну комірку прописуємо
20  значення елементу масиву за його номером */
21  {
22  echo"<table border=1> <tr align=center>
23  <td width=20>$items[0]</td>
24  <td width=60>$items[1]</td>
25  <td width=500>$items[2]</td>
26  <td width=100>$items[3]</td>
27  <td width=80>$items[4]</td>
28  <td width=80>$items[5]</td>
29  </tr></table>";
30  }
31  ?>
32  </body>
33  </html>
```

Рисунок 6.16 – Вміст скрипту view_news.php

Вміст звіту

а) тема і мета лабораторної роботи;
б) у відповідності до кроків ходу роботи навести:
   1) графічне представлення інженерних дій у вигляді схем програм, схем взаємодії програмних модулів, діаграм використання або інших UML-діаграм;
   2) лістинг всіх відлагоджених програмних модулів;
   3) результати виконання програмних модулів;
в) змістовні висновки за результатами роботи.

Питання для самоконтролю

1. Які види числових типів застосовують в технології PHP-MySQL?
2. Який діапазон значень типу SMALLINT[(max)]?
3. Які види стрічкових типів застосовують в технології PHP-MySQL?
4. У якому форматі задається дата і час типу DATETIME?
5. Яке призначення функції CREATE TABLE ()?
6. Яке призначення функції ALTER TABLE ()?
7. За допомогою якого метода функції ALTER TABLE() можна змінити поле?
8. За допомогою якого оператора передається значення полів?
9. Для чого призначена команда UPDATE?
10. Наведіть приклад застосовування оператора умови WHERE?
11. Для чого застосовується оператор умови SELECT?
12. Наведіть приклад застосування оператора mysql_fetch_row?
13. Наведіть приклад запиту DELETE у зроблених вами скриптів?
14. Що таке первинний ключ?

ЛАБОРАТОРНА РОБОТА №7. ДОСЛІДЖЕННЯ ОПЕРАТОРІВ ДЛЯ РОБОТИ З РЯДКОВИМИ ЗМІННИМИ ТА РЕГУЛЯРНИМИ ВИРАЗАМИ

Мета: набуття навичок у використанні рядкових змінних та регулярних виразів для побудови складних WEB-проектів.

Теоретичні відомості

Тільки нещодавно усталене в міжнародній спільноті поняття електронного контенту (ЕК) набуває помітного соціального та, навіть, економічного значення. Аналіз ЕК є надзвичайно важливою проблематикою, що підтверджують нові тенденції побудови Semantic WEB у відповідності до стандартів міжнародного консорціуму W3C. Оскільки природномовна складова грає найважливішу роль в усьому ЕК, оволодіння навичками роботи з рядковими змінними та регулярними виразами надзвичайно актуальне для практики розробників складних WEB-проектів, де передбачений аналіз контенту.

Довідник всіх функцій мови PHP на російській мові доступний за адресою http://ru2.php.net/manual/ru/reference.pcre.pattern.syntax.php. Опис рядкових функцій та регулярних виразів також відображено у популярному підручнику http://www.softtime.ru/bookphp/gl7_12.php.

Уміння грамотно працювати з рядками дуже важливо для програміста. Рядкових функцій в PHP чимало, причому багато з них, так чи інакше, просто дублюють один одного. Однак, прийоми роботи з основними блоками функцій (функції пошуку і заміни в тексті, функції видалення пробільних символом (trim-функції), функції форматного виведення) треба представляти дуже чітко.

Взагалі кажучи, за визначенням, рядок – це послідовність символів типу char, що закінчується нульовим символом. Тому, можна, звичайно, радіти тому, що при роботі з рядками не доводиться жонглювати покажчиками в стилі C/C++, проте не варто забувати про те, що ціна такій зручності – швидкодія. Тому наведені функції потрібно обговорювати, в тому числі, і з точки зору їх швидкодії.

Традиційно для пошуку в рядку підрядка використовуються строкові функції, обмежуючись при цьому випадками точної відповідності рядків і підрядків. Для роботи з рядками у більш складних випадках доцільно використовувати також регулярні вирази, що представляють собою, по суті, міні-мову описів для пошуку в рядках інформації за заданим шаблоном.

За допомогою регулярних виразів можна знайти в рядку підрядок, що задовольняє заданому шаблоном і витягти його, перевірити чи існує заданий рядок тощо.

Алгоритм пошуку з використанням регулярних виразів був вперше розроблений одним із творців UNIX Кеном Томпсоном. Цікаво, що спочатку регулярні вирази з'явилися не в теорії обчислювальних систем, а в нейрофізіології. Основу теорії регулярних виразів заклали нейрофізіологи У. Мак-Каллох і У. Піттс, що працювали над способами математичного опису нервових процесів. Пізніше математик С. Кліні, ґрунтуючись на цих дослідженнях, опублікував роботу "Представлення подій в нейронних мережах", в якій і було введено поняття регулярних виразів. А вже Кен Томпсон, ґрунтуючись на роботах Кліні, адаптував теорію регулярних виразів для алгоритмів пошуку інформації. Саме починаючи з його робіт, регулярні вирази стали використовуватися в текстових редакторах і увійшли в багато мов програмування.

Базовий синтаксис і створення регулярних виразів

Найпростіший регулярний вираз можна записати так: "abc". Цей вираз відповідає будь-якому рядку, що містить підрядок "abc".

Існує таке поняття, як вираз у квадратних дужках. Квадратні дужки обмежують пошук тими символами, які в них укладені: "[abc]". Цьому регулярному виразу відповідає будь-який рядок, що містить символи abc або разом, або кожний з них окремо. Вираз у квадратних дужках іноді ще називають бракет-виразом.

Припустимо, що нам потрібно створити регулярний вираз, що відповідає всім буквам російського алфавіту. У цьому випадку ми можемо, звичайно, перелічити всі ці букви в регулярному виразі. Це допустимо, але утомливо і неелегантно. Тому значно лаконічніше такий регулярний вираз можна записати як: "[а-Я]".

Останній вираз відповідає всім буквам російського алфавіту, оскільки будь-які два символи, що розділяються дефісом, задають відповідність діапазону символів, що знаходяться між ними. Зауважте, що регулярний вираз "[а-Я]" описує символи як нижнього, так і верхнього регістрів, тому більш детально цей вираз можна записати так: "[а-яА-Я]".

Точно таким же чином задаються регулярні вирази, відповідні числам: "[0-9]" або "[0123456789]". Обидва цих вирази еквівалентні і відповідають будь-якій цифрі.

Для виключення послідовності символів з пошуку перед цією ній ставиться символ "^": "[^ а-я]". Цей регулярний вираз відповідає будь-якому символу, що не міститься у діапазоні а-я. Зверніть увагу – символ ^ знаходиться всередині квадратних дужок, так як тільки у цьому випадку він має значення "не".

У регулярному виразі може бути кілька гілок, які розділяються символом |, чинним як оператор OR (АБО). Тобто, якщо у виразі використовуються гілки, то для відповідності регулярного виразу деякому рядку достатньо, щоб тільки одна з гілок відповідала цьому рядку: "abc | АБВ". Цьому регулярному виразу відповідає будь-який рядок, що містить підрядок "abc"

або "АБВ". Розгалуження зручно застосовувати при перевірці розширень і імен файлів, зон доменних імен тощо. Наприклад, наступне регулярний вираз перевіряє, чи містяться в рядку підрядка "ru", "com" або "net": "ru | com | net".

Регулярний вираз можна уточнити за допомогою кваліфікаторів – так називаються символи +, ?, *. Кваліфікатори говорять про те, скільки разів послідовність символів може зустрітися в рядку і вміщуються безпосередньо після тієї частини виразу, до якої вони застосовуються:

а) "а +" - хоча б один а (рядки "АБВ" і "абва" відповідають цьому виразу, а рядка "укр" - немає);

б) "а?" - Нуль або один а (рядки "АБВ" і "укр" відповідають цьому виразу, а рядка "абва" - немає);

в) "а *" - нуль або більше а (рядки "АБВ" і "абва" і "укр" відповідають цьому виразу). Цей регулярний вираз відповідає будь-якому символу, що не міститься у діапазоні а-я.

Підходи та стандарти

На сьогодні існує декілька альтернативних підходів та відповідних стандартів для реалізації регулярних виразів. PHP підтримує два види запису регулярних виразів: POSIX і Perl. POSIX розшифровується як Portable Operating System Interface (інтерфейс мобільної операційної системи) і є стандартом для інтерфейсів додатків. Всі вище розглянуті можливості базового синтаксису походять від стандарту POSIX, але він натепер (PHP 5.3.0 і вище) вже вважається застарілим.

Perl-зумісний стандарт регулярних виразів має назву PCRE (Perl Compatible Regular Expression) і найбільш популярною функцією цього стандарту є preg_match() на відміну від аналогічної ereg() від POSIX. Вирази стандарту PCRE відрізняються обмежувачами, в якості яких, як правило, використовується прямий слеш: / mail /. Цей вираз відповідає рядку, що містить слово "mail".

За обмежувачами можуть слідувати модифікатори, уточнюючі регулярний вираз. Найбільш часто використовують наступні модифікатори:

і (Ігнорувати регістр при пошуку відповідності, тобто вираз "/ mail / і" буде відповідати як рядку "MAIL", так і рядку "mail");

х (Ігнорувати пробільні символи).

Для PCRE-виразів характерне застосування загальних типів символів, які аналогічні класам символів для регулярних виразів у стилі POSIX. Найбільш часто зустрічаються загальні типи символів наведені нижче:

- \ d - відповідність десятковій цифрі;
- \ D - відповідність будь-якому символу, за винятком десяткових цифр;
- \ s - відповідність пробільним символам;
- \ S - відповідність будь-яким символам, окрім пробільних;
- \ h - наявність межі слова;

- \ B - відсутність межі слова;
- \ Z - кінець рядка або символ перекладу кінця рядка;
- \ A - початок рядка.

У межах нових стандартів Semantic WEB консорціум W3C запропонував об'єктну модель документу (DOM) як розвиток мови XML. Фактично DOM – це інтерфейс для роботи з структурою XML-документів. Коли ви розбираєте XML-документ за допомогою парсеру DOM, ви отримуєте ієрархічну структуру даних (дерево DOM), яка відповідає всьому тому, що парсер знайшов в XML-документі. Ви можете потім використовувати функції DOM, щоб маніпулювати деревом, переміщати гілки, додавати нові гілки, видаляти частини дерева. Докладно про можливості та функції DOM можна подивитися безпосередньо за адресою консорціуму http://www.w3.org/DOM/ .

З метою порівняння двох сучасних підходів пропонується розглянути та проаналізувати два програмних коди, які можна використати для виконання завдання даної лабораторної роботи:

а) обробка сторінки за допомогою функції preg_match() (рисунок 7.1):

```
1   parser_regex.php
2   <html>
3   <head>
4   <title>Parser</title>
5   <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
6   <link rel="stylesheet" type="text/css" href="style.css">
7   </head>
8   <body>
9   <div id="main">
10  <?
11  // підключаємо файл з налаштуваннями з'єднання
12  include_once("config.php");
13  // отримуємо вхідні параметри
14  $url = $_POST['url']; // посилання на сторінку зі зображеннями
15  $depth = $_POST['depth']; // кількість сторінок для обробки
16  // створюєм з'єднання з базою даних
17  $db = new PDO("mysql:host=$host;dbname=$db_name", $user, $pass);
18  // перевіряєм параметри
19  if($url != '') {
20  if($depth == '') { // якщо кількість сторінок не задана - обробляєм лише
21  одну сторінку
22  parsePage($url, 1, $db);
23  } else {
24  parsePage($url, $depth, $db);
```

Рисунок 7.1 – Обробка сторінки за допомогою функції preg_match(), аркуш – 1

```
25  }
26  }
27  echo '<p align="center"><a href="search.php">Search</a></p>';
28  // рекурсивна функція пошуку посилань на зображення та запису їх в базу даних
29  function parsePage($url, $depth, $db) {
30  if ($depth == 0) return 0;
31  // завантажуємо сторінку за заданим посиланням
32  $html = file_get_contents($url);
33   // виводимо на сторінку поточну категорію зображень
34  $category = getCategory($html);
35  echo "<h6>".$category."</h6>";
36  // отримуємо номер поточної сторінки та виводимо його
37  $nPage = getPageNum($html);
38  echo "<b>Page: ".$nPage."</b><br/>";
39  /*
40  * За допомогою регулярного виразу знаходимо всі блоки "div" які містять
41  необхідні зображення.
42  * Всі вони мають атрибут "class" зі значенням "foto".
43  */
44  preg_match_all("/<div class='foto'>([^\"]+)/", $html, $divs);
45  for ($i = 0; $i < count($divs[0]); ++$i) {
46  // отримуємо атрибут "src" зображення
47  preg_match("/src=[\"'](.+?)[\"']/", $divs[1][$i], $srcs);
48  // отримуємо атрибут "alt" зображення
49  preg_match("/alt='([^\"]+)' /", $divs[1][$i], $alts);
50  $src = $srcs[1];
51  // відкидаємо зайвий текст та зберігаємо назву зображення
52  $title = substr(substr($alts[1], 4), 0 , -4);
53  // виводимо посилання на зображення
54  echo "<a href='$src'>$title</a><br/>";
55  // зберігаємо дані про зображення в таблиці Pictures
56  $row = $db->exec("INSERT INTO `Pictures` VALUES (null, '$title',
57  '$category', '$src')");
58  // отримуємо id зображення в таблиці "Pictures"
59  $last_ins_id = $db->lastInsertId();
60  // за допомогою регулярного виразу ділимо строку з назвою зображення на слова
61  $words = preg_split("/\s+/s", $title);
62  // кожне слово зберігаємо в таблицю Words
63  // де поле pID це id зображення
64  for($j=0;$j<count($words);++$j) {
65  if(strlen($words[$j]) > 0)
66  $db->exec("INSERT INTO `Words` VALUES (null,
67  '$words[$j]', '$last_ins_id')");
68  }
69  }
70  // отримуємо номер наступної сторінки
71  $nPage--;
```

Рисунок 7.1 – Обробка сторінки за допомогою функції preg_match(), аркуш - 2

```
72  $depth--;
73  if($nPage == 0) return 0;
74  // отримуємо URL наступної сторінки
75  $nextPage = str_replace("htm", "stranica".$nPage.".htm", $url);
76  // викликаєм функцію parsePage для обробки наступної сторінки
77  parsePage($nextPage, $depth, $db);
78  }
79  // функція для знаходження номера поточної сторінки
80  function getPageNum($html){
81  // отримуємо список всіх елементів з тегом "font"
82  $result = preg_match("/<font color=red>[0-9]+/", $html, $pages);
83  // за допомогою регулярного виразу фільтруємо строку відкидаючи всі
 84   символи крім цифр
85  preg_match("|\d+|", $pages[0], $nPages);
86  // $nPages[0] містить номер сторінки
87  return $nPages[0];
88  }
89  // функція для знаходження поточної категорії
90  function getCategory($html){
91  // за допомогою регулярного виразу отримуємо вміст тега "title"
92  $result = preg_match("/<title>(?<=^|>)[^><]+?(?=<|$)/", $html, $div);
93  // назва категорії знаходиться між символами ":" та "-"
94  // визначимо їх позиції в строці відповідно
95  $pos1 = strpos($div[0], ':');
96  $pos2 = strpos($div[0], '-');
97  // видалимо зі строки символи після символа "-" включно
98  $cat = substr($div[0], 0, $pos2);
99  // видалимо зі строки символи до символа ":" включно
100 return substr($cat, $pos1+1, $pos2);
101 }
102 ?>
103 </div>
104 </body>
105 </html>
```

Рисунок 7.1 – Обробка сторінки за допомогою функції preg_match(), аркуш - 3

б) обробка сторінки за допомогою DOM (рисунок 7.2):

```
1   $url = $_POST['url']; // посилання на сторінку зі зображеннями
2   $depth = $_POST['depth']; // кількість сторінок для обробки
3   // створюєм з'єднання з базою даних
4   $db = new PDO("mysql:host=$host;dbname=$db_name", $user, $pass);
5   // перевіряєм параметри
6   if($url != '') {
7   if($depth == ''){//якщо кількість сторінок не задана - обробляєм лише одну сторінку
```

Рисунок 7.2 – Обробка сторінки за допомогою DOM, аркуш - 1

```
8   parsePage($url, 1, $db);
9   } else {
10  parsePage($url, $depth, $db);
11  }
12  }
13  echo '<p align="center"><a href="search.php">Search</a></p>';
14  // функція пошуку посилань на зображення та запису їх в базу даних
15  function parsePage($url, $depth, $db){
16  // завантажуємо сторінку за заданим посиланням
17  $html = file_get_contents($url);
18  // створюємо новий DOM-документ
19  $dom = new DOMDocument();
20  // завантажуємо в нього сторінку
21  @$dom->loadHTML($html);
22  // створюємо об'єкт для пошуку необхідних елементів
23  $finder = new DomXPath($dom);
24  // виводимо на сторінку поточну категорію зображень
25  $category = getCategory($dom, $finder);
26  echo "<br/>Category: ".$category."<br/>";
27  // отримуємо номер поточної сторінки
28  $nPage = getLastPageNumber($dom);
29  for($i = 0; $i < $depth; ++$i) {
30  echo "<b><br/>Page: ".$nPage."</b><br/>";
31  $imageClassName="foto";
32  /*
33  * Знаходимо всі блоки "div" які містять необхідні зображення.
34  * Всі вони мають атрибут "class" зі значенням "foto".
35  * $nodes - масив з блоками
36  */
37  $nodes = $finder->query("//*[contains(concat(' ',
38  normalize-space(@class), ' '), '
39  $imageClassName ')]");
40  for ($ii = 0; $ii < $nodes->length; $ii++) {
41  // отримуємо елемент масиву
42  $node = $nodes->item($ii);
43  // отримуємо список елементів з тегом "img"
44  $list = $node->getElementsByTagName('img');
45  // отримуємо атрибут "src" першого елементу списка, оскільки
46  елемент лише один
47  $src = $list->item(0)->getAttribute('src');
48  // отримуємо атрибут "alt" першого елементу списка
49  $alt = $list->item(0)->getAttribute('alt');
50  // відкидаємо зайвий текст та зберігаємо назву зображення
51  $title = substr(substr($alt, 8), 0 , -8);
52  // виводимо посилання на зображення
53  echo "<a href='$src'>$title</a><br/>";
54  // зберігаємо дані про зображення в таблиці Pictures
55  $row = $db->exec("INSERT INTO `Pictures` VALUES (null, '$title',
56  '$category', '$src')");
```

Рисунок 7.2 – Обробка сторінки за допомогою DOM, аркуш - 2

```
57  // отримуємо id зображення в таблиці "Pictures"
58  $last_ins_id = $db->lastInsertId();
59  // ділимо строку з назвою зображення на слова
60  $words = explode(" ", $title);
61  // кожне слово зберігаємо в таблицю Words
62  // де поле pID це id зображення
63  for($j=0;$j<count($words);++$j) {
64  if(strlen($words[$j]) > 0)
65  $db->exec("INSERT INTO `Words` VALUES (null,
66  '$words[$j]', '$last_ins_id')");
67  }
68  }
69  // якщо задана кількість сторінок оброблена - завершуємо функцію
70  if( $i == $depth-1 ) break;
71  // отримуємо номер наступної сторінки
72  $nPage--;
73  // отримуємо URL наступної сторінки
74  $nextPage = str_replace("htm", "stranica".$nPage.".htm", $url);
75  // завантажуємо сторінку за заданим посиланням
76  $html = loadPage($nextPage);
77  // створюємо новий DOM-документ
78  $dom = new DOMDocument();
79  // завантажуємо в нього сторінку
80  @$dom->loadHTML($html);
81  // створюємо _____об'єкт для пошуку необхідних елементів
82  $finder = new DomXPath($dom);
83  }
84  }
85  // функція для знаходження поточної категорії
86  function getCategory($dom, $finder){
87  $categoryClassName="ftext";
88  /*
89  * Знаходимо всі блоки "div" які містять необхідний текст.
90  * Всі вони мають атрибут "class" зі значенням "ftext".
91  * $cNodes - масив з блоками
92  */
93  $cNodes = $finder->query("//*[contains(concat(' ', normalizespace(@
94  class), ' '), ' $categoryClassName ')]");
95  // отримуємо перший і єдиний блок
96  $cNode = $cNodes->item(0);
97  // отрумуємо внутрішній блок з посиланнями
98  $cList = $cNode->getElementsByTagName('div');
99  // отримуємо список всіх посилань
100 $linksList = $cList->item(0)->getElementsByTagName('a');
101 // необхідний текст в останньому посиланні
102 return $linksList->item($linksList->length - 1)->textContent;
103 }
104 // функція для знаходження номера поточної сторінки
105 function getLastPageNumber($dom){
```

Рисунок 7.2 – Обробка сторінки за допомогою DOM, аркуш - 3

```
106  // отримуємо список всіх елементів з тегом "font"
107  $pList = $dom->getElementsByTagName('font');
108  $nPages = 0;
109  foreach ($pList as $p) {
110  // знаходимо елемент з номером сторінки
111  if(strstr($p->nodeValue, '(самые новые')){
112  // за допомогою регулярного виразу фільтруємо строку відкидаючи
113  всі символи крім цифр
114  preg_match("|\d+|", $p->nodeValue, $nPages);
115  }
116  }
117  // $nPages[0] містить номер сторінки
118  return $nPages[0];
119  }
120  ?>
121  </div>
122  </body>
123  </html>
```

Рисунок 7.2 – Обробка сторінки за допомогою DOM, аркуш – 4

Хід роботи

1. Сформувати html сторінку з формою ведденя даних
2. Підключення бази даних з інформацією(id,e-mail,tel,data).
3. Розробити динамічну сторінку, що використовує форми і вікна як інтерфейсні елементи.
4. Створити інтерактивну форму.
5. Перевірте довільний рядок, чи є він правильним e-mail адресом. Використовуйте функцію preg_match.
6. Створити у відповідь на подію вікно з текстом.
7. Напишіть функцію, яка із введеного тексту знаходить дати в різних форматах.
8. Перевірте телефонні номера (за шаблоном +38 (097) 65-94-888).
9. Є масив дат у різних форматах (рррр-мм-дд, дд-мм-рррр). Приведіть їх до формату мм.дд.рррр.
10. Вивід на екран інформації з БД.

Побудова складних Web-проектів

Для початку роботи створюємо сторінку з формою введення даних.

```
<form action="" method="post">
E-Mail: <input type="text" name="email"/><br/>
Телефон: <input type="text" name="tel"/><br/>
Дата: <input type="date" name="date"/><br/>
<input type="submit" name="send" value="Send"/>
</form>';
```

Рисунок 7.3 – Створення форми введення даних

Зчитування інформації з бази даних відбувається наступним чином:

```
function DateInput($Date)
{
    $date_explode=explode(".", $Date);
    $result_date = $date_explode[0].'.'.$date_explode[1].'.'.$date_explode[2];
    return $result_date;
}
```

Рисунок 7.4 – Зчитування інформації з бази даних

Перевірка на правильність e-mail адреси, телефона здійснюється з використанням функції preg_match. Аргументами функції є елементи e-mail адреси розміщені в правильному порядку, якщо адреса введена не вірно то після перевірки відбувається виведення на екран віконця про помилку:

```
    if(!preg_match("/^([a-z0-9_\.-]+)@([a-z0-9_\.-]+)\.([a-z\.]{2,6})$/ui",
$_POST['email']))
    {
        echo '<script language="JavaScript">alert("Невірний Email!");</script>';
    }
    else if(!preg_match("/^\+380([0-9]{9})$/ui", $_POST['tel']))
    {
        echo '<script language="JavaScript">alert("Невірний телефон!");</script>';
    }
    else
        mysql_query('INSERT INTO `users_data` (`date`, `tel`, `email`) VALUES
('".DateInput($_POST['date'])."', '".$_POST['tel']."', '".$_POST['email']."')');
    //echo TagCut($_POST['text'], 'i');
```

Рисунок 7.5 – Перевірка на правильність введення даних

Для знаходженя дати з вказаного тексту необхідно використати функцію preg_match_all. Першим аргументом функції є регулярний вираз, другим аргументом зміна з якої відбувається зчитування тексту і третім аргументом є масив даних в який відбувається запис даних.

```
1   if(isset($_POST['ok']))
2   {
3       $date = $_POST['email'];
4       preg_match_all('/((\d{4})-(\d{2})-(\d{2}))|((\d{2})-(\d{2})-(\d{4}))/', $date,
5   $date_parts);
6       foreach($date_parts[0] as $Value)
7       echo DateEdit($Value).'<br/>';
8   }
9
10 echo '<form action="" method="post">
11 <textarea name="email" style="width:200px;height:140px;"> оот аутао 09-02-2013 аьлал
12 ука уак 2002-09-12hfghshsdfgh2002-11-02</textarea><br/>
13 <input type="submit" name="ok" value="Send"/>
14 </form>';
```

Рисунок 7.6 – Знаходженя дати з вказаного тексту

Для форматування дати виконуємо перестановку елемнтів масиву.

```
1   function DateEdit($Date)
2   {
3       $date_explode=explode("-", $Date);
4       if($date_explode[0] > 31)
5       {
6               $day= $date_explode[2];
7               $year = $date_explode[0];
8       }
9       else
10      {
11              $day= $date_explode[0];
12              $year = $date_explode[2];
13      }
14      $result_date = $day.'.'.$date_explode[1].'.'.$year;
15      return $result_date;
16  }
```

Рисунок 7.7 – Форматування дати

Вміст звіту

а) тема і мета лабораторної роботи;
б) у відповідності до кроків ходу роботи навести:
   1) графічне представлення інженерних дій у вигляді схем програм, схем взаємодії програмних модулів, діаграм використання або інших UML-діаграм;
   2) лістинг всіх відлагоджених програмних модулів;
   3) результати виконання програмних модулів;
в) змістовні висновки за результатами роботи.

Питання для самоконтролю

1. Що таке W3C?
2. Які тенденції побудови Semantic WEB вам відомі?
3. Які рядкові функції підтримує мова PHP?
4. Що таке регулярний вираз?
5. Що працює швидше – рядкова функція чи аналогічний регулярний вираз?
6. Як створювалася теорія регулярних виразів?
7. Що вважається найпростішим регулярним виразом?
8. Що таке вираз у квадратних дужках?
9. Яке значення має дефіс у регулярному виразі?
10. Яке значення має символ ^ у регулярному виразі?
11. Яким чином реалізовані розгалуження у регулярних виразах?
12. Що таке кваліфікатор?
13. Охарактеризуйте стандарт POSIX для написання регулярних виразів?
14. Охарактеризуйте стандарт PCRE для написання регулярних виразів?
15. У чому відмінності між стандартами POSIX та PCRE?
16. Охарактеризуйте об’єктну модель документу (DOM) з точки зору побудови регулярних виразів?
17. Які основні функції підтримує DOM?

ЛАБОРАТОРНА РОБОТА №8. ПРОЕКТУВАННЯ ТА РОЗРОБКА WEB-САЙТІВ

Мета: ознайомлення з сучасними фреймворками та шаблоном проектування WEB-сайтів MVC (Модель–Вид–Контроллер), одержання навичок у розробці типових WEB-форм за допомогою фреймворку Codeignaiter.

Теоретичні відомості

Створення сучасного web-сайту з відповідними додатками являє собою досить трудомістку задачу. Коли ви читаєте книгу або керівництво, то все здається досить простим. Демонстраційні програми рідко бувають більше декількох сторінок.

Проблема полягає у тому, що це лише учбові приклади. Як тільки ви спробуєте використовувати їх на практиці, ви побачите, як збільшується обсяг коду. Розглянемо обробку даних, отриманих з форми. Ми повинні:

- перевірити тип отриманих значень;
- переконатися, що значення мають допустимі величини (або формат);
- видалити з тексту заборонені теги (захист від XSS);
- замінити в параметрах SQL-запитів службові символи на їх ескейп послідовності (захист від SQL Injection);
- якщо параметри пов'язані між собою, перевірити ці зв'язки.

Природно, цей код повинен бути не тільки написаний, але і протестований. Як бачите, пристойний шматок роботи. Причому в більшості своїй це рутинні операції, які повторюються практично в усіх програмах.

Отже, щоб не повторювати одне й те саме, сьогодні для створення web сайту (додатки) існує три основні підходи (або їх комбінації):

1) Використовувати «чистий» PHP + стандартні та додаткові бібліотеки. Цей варіант найбільш трудомісткий, але при цьому володіє найбільшою гнучкістю. Ви можете реалізувати практично будь-який функціонал і при цьому забезпечити максимальну продуктивність. Правда є один нюанс. Хороший продукт ви отримаєте тільки після тестування та оптимізації програми, а це зовсім не такий простий процес, як здається на перший погляд.

2) Використовувати готове рішення. На сьогоднішній день практично для всіх широко використовуваних типів сайтів існують готові движки. Наприклад, WordPress, Joomla, PHPbb та багато інших. Тут можна взагалі обійтися без програмування, тому що ці рішення являють собою готові сайти (блоги, портали, форуми тощо) ви тільки створюєте контент. Звідси і назва CMS (content management system) - системи управління контентом. Якщо стандартної функціональності не вистачає, то можна написати плагін (або знайти готовий).

В принципі багато таких систем забезпечують непогану продуктивність, але тільки в тих завданнях, для яких вони спочатку проектувалися. Тобто ви зможете додати потрібні вам функції, але при цьому продуктивність (споживання ресурсів) може бути значно гірше, ніж у першому варіанті.

3) Використовувати фреймворк (framework). В принципі, фреймворк можна вважати додатковою бібліотекою. Але є суттєва відмінність.

Бібліотеку ви використовуєте для розширення функціональності програми. А фреймворк крім того визначає архітектуру (взаємозв'язку між компонентами) програми. В принципі, використання фреймворка це щось середнє між першим і другим варіантом. З одного боку свобода ваших дій буде обмежена в порівнянні з першим варіантом, але ці обмеження незначні в порівнянні з готовими рішеннями.

Детальніше розглянемо третій варіант.

Фреймворк (Framework) – програмний каркас, призначений для розробки в веб-сайтів, веб-додатків або веб-служб.

Фреймворк полегшує процес розробки сайту, надаючи свої інструменти для різних завдань таких як: бібліотеки для доступу до баз даних, шаблонізатор, управління сесіями і багато інших, також сприяє повторному використанню коду.

На відміну від бібліотек, які об'єднують набір підпрограм близької функціональності, фреймворк містить в собі велику кількість різних за призначенням бібліотек. Може включати допоміжні програми, бібліотеки коду, мову сценаріїв та інше програмне забезпечення, що полегшує розробку та об'єднання різних компонентів великого програмного проекту. Зазвичай об'єднання відбувається за рахунок використання єдиного API (прикладного програмного інтерфейсу).

Прикладами можуть бути системи управління контентом (CMS), веб-фреймворки.

Фреймворк визначається як безліч конкретних і абстрактних класів, а також визначень способів їх взаємодії. Конкретні класи зазвичай реалізують взаємовідносини між класами. Абстрактні класи являють собою точки розширення, в яких каркаси можуть бути використані або адаптовані.

Точка розширення – це та частина фреймворка, для якого не наведена реалізація. Відповідно каркас концептуальної моделі складається з концептуальних класів, а каркас програмної системи з класів мови програмування загального призначення.

Найвідоміші веб-фреймворки представлені в таблиці 8.1.

Таблиця 8.1 – Найвідоміші веб-фреймворки

Емблема	Назва	Короткий опис
	**Yii**	Високоефективний фреймворк, заснований на базі компонентної структури PHP для розробки великих веб-додатків, який дозволяє максимально використовувати концепцію повторного застосування коду і може значно прискорити весь процес веб-розробки. Сама назва Yii вимовляється як Yee і означає простий (easy), розширюваний (extensible) і ефективний (efficient).
	**Symfony**	Фреймворк з повністю відкритим кодом, написаний на PHP5. Використання цього фреймворку дозволяє створювати читабельний структурований код і набагато спрощує програмування, так як величезні набори складних операцій досить часто можна замінити лише одним рядком коду.
	**Kohana**	Об'єктнорієнтований HMVC веб-фреймворк з відкритим вихідним кодом, написаним на PHP5. Kohana поширюється тільки за ліцензійною угодою BSD, що надає вам повне право легально застосовувати її під будь-які комерційні проекти.
	**CodeIgniter**	Всім відомий фрейморк для розробки величезної кількості веб-додатків, що використовує PHP як мову програмування. Головна його мета полягає в тому, щоб допомогти вам розробляти ітернет-проекти набагато швидше, ніж ви б змогли, вводячи весь код самому.
	**Zend Framework**	Складна бібліотека класів, за допомогою якої розробляються web-додатки. Необхідно звернути увагу на те, що застосування бібліотек класів значною мірою зменшує терміни написання програмних продуктів, завдяки раніше спроектованому і написаному сценарію. Творці Zend Framework спроектували велику кількість класів, які дають можливість здійснювати різноманітні завдання, що ставляться перед командою web-програмістів.

Більш повну таблицю можна знайти за посиланням: http://rmcreative.ru /playground/php-frameworks/

Шаблони проектування Модель – Вид – Контролер

Model-view-controller (Модель-вид-контролер) – схема використання декількох шаблонів проектування, за допомогою яких модель даних програми, користувальницький інтерфейс і взаємодія з користувачем розділені на три окремих компонента так, що модифікація одного з компонентів надає мінімальний вплив на інші. Дана схема проектування часто використовується для побудови архітектурного каркаса, коли переходять від теорії до реалізації в конкретній предметній області. Схема концепції шаблону наведена на рисунку 8.1.

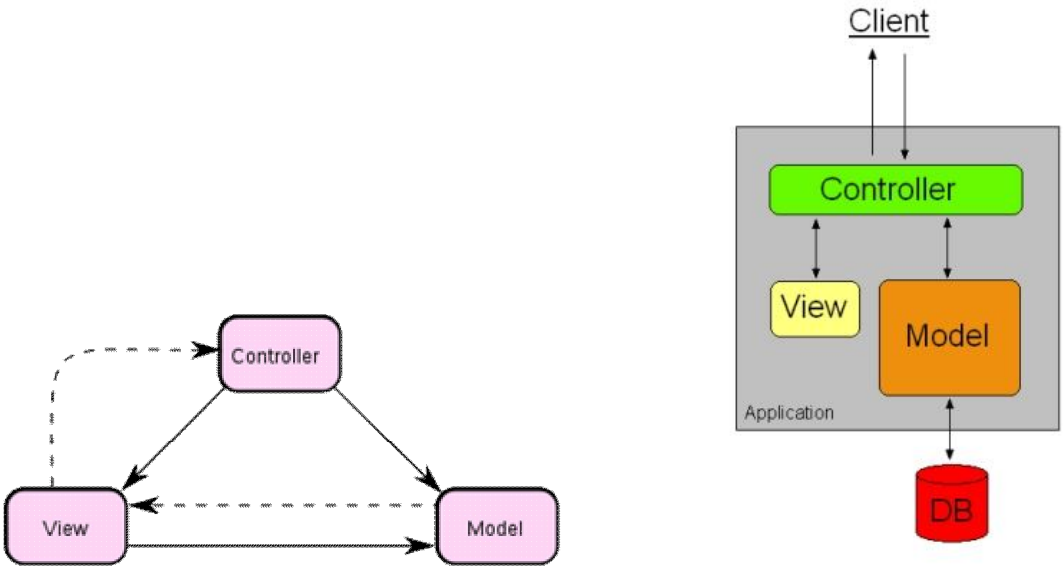

Рисунок 8.1 – Концепція Model-View-Controller

Суцільними лініями показані прямі зв'язки (виклики методів, присвоєння значень полів), переривчастими лініями показані непрямі зв'язку (повідомлення через події).

Концепція MVC була описана в 1979 році Трігве Реенскаугом (англ. Trygve Reenskaug), тоді працюючому над мовою програмування Smalltalk в Xerox PARC. Оригінальна реалізація описана в статті «Applications Programming in Smalltalk-80: How to use Model-View-Controller». Потім Джим Алтофф з командою розробників реалізували версію MVC для бібліотеки класів Smalltalk-80.

В оригінальній концепції була описана сама ідея і роль кожного з елементів моделі, виду та контролера. Але зв'язки між ними були описані без конкретизації. Крім того, розрізняли дві основні модифікації:

1. Пасивна модель – така, яка не має ніяких способів впливати на подання або контролер і використовується ними як джерело даних для відображення. Всі зміни моделі відслідковуються контролером і він же відповідає за перемальовування подання, якщо це необхідно. Пасивна модель

частіше використовується в структурному програмуванні, так як в цьому випадку є просто структурою даних без методів, які їх обробляють.

2. Активна модель – така, яка оповіщає вигляд про те, що в ній відбулися зміни, а вигляд, який зацікавлений в оповіщенні, підписується на ці повідомлення. Це дозволяє зберегти незалежність моделі як від контролера, так і від вигляду.

Класичною реалізацією паттерна MVC прийнято вважати версію саме з активною моделлю.

З розвитком об'єктно-орієнтованого програмування і поняття про шаблони проектування було створено ряд модифікацій концепції MVC, які при реалізації у різних авторів можуть відрізнятися від оригінальної. Так, наприклад, Еріан Вермі в 2004 році описав приклад узагальненого MVC.

Призначення

Основна мета застосування цієї концепції полягає в поділі бізнес-логіки (моделі) від її візуалізації (подання, виду). За рахунок такого поділу підвищується можливість повторного використання. Найбільш корисне застосування даної концепції коли користувач повинен бачити ті ж самі дані одночасно в різних контекстах і(або) з різних точок зору. Зокрема, виконуються наступні завдання:

1. До однієї моделі можна приєднати кілька виглядів, при цьому не зачіпаючи реалізацію моделі. Наприклад, деякі дані можуть бути одночасно представлені у вигляді електронної таблиці, гістограми і кругової діаграми.

2. Не торкаючись реалізацію виглядів, можна змінити реакції на дії користувача (натискання мишею на кнопці, введення даних), для цього досить використовувати інший контролер.

3. Ряд розробників спеціалізуються тільки в одній з областей: або розробляють графічний інтерфейс або розробляють бізнес-логіку. Тому можливо добитися, що програмісти, які займаються розробкою бізнес-логіки (моделі), взагалі не будуть обізнані про те, який вигляд буде використовуватися.

Концепція

Концепція MVC дозволяє розділити дані, вигляд та обробку дій користувача на три окремі компоненти:

- модель (англ. Model). Модель надає знання: дані і методи роботи з цими даними, реагує на запити, змінюючи свій стан. Не містить інформації, як ці знання можна візуалізувати;
- представлення, вигляд (англ. View). Відповідає за відображення інформації (візуалізація). Часто як уявлення виступає форма (вікно) з графічними елементами;
- контролер (англ. Controller). Забезпечує зв'язок між користувачем і системою: контролює введення даних користувачем і використовує модель та представлення для реалізації необхідної реакції.

Важливо відзначити, що як вигляд, так і контролер залежать від моделі. Однак модель не залежить ні від представлення, ні від контролера. Тим самим досягається призначення такого поділу: воно дозволяє будувати модель незалежно від візуального представлення, а також створювати кілька різних представлень для однієї моделі.

Для реалізації схеми Model-View-Controller використовується досить велика кількість шаблонів проектування (в залежності від складності архітектурної рішення), основні з яких Спостерігач, Стратегія, Компонувальник.

Найбільш типова реалізація відокремлює представлення від моделі, шляхом встановлення між ними протоколу взаємодії, використовуючи апарат подій (підписка / оповіщення). При кожній зміні внутрішніх даних в моделі, модель оповіщає всіх залежних від неї представлень, і вони оновлюється. Для цього використовується шаблон проектування спостерігач. При обробці реакції користувача, представлення вибирає залежно від потрібної реакції потрібний контролер, який забезпечить той чи інший зв'язок з моделлю. Для цього використовується шаблон проектування стратегія, або замість цього може бути модифікація з використанням шаблону проектування команда. А для можливості однотипного поводження з під-об'єкта складового ієрархічного виду, може використовуватися шаблон проектування Компонувальник. Крім того, можуть використовуватися й інші шаблони проектування, наприклад, фабричний метод, який дозволить задати за замовчуванням тип контролера для відповідного представлення.

Отже, ми отримали найпростішу MVC-систему. Виділимо позитивні і негативні сторони. До мінусів можна віднести:

- збільшення обсягу коду;
- необхідність дотримання заздалегідь заданого інтерфейсу;
- для підтримки розробки потрібні більш кваліфіковані фахівці.

Остання вимога до нашого прикладу не відноситься, але для реальних систем воно дуже актуально.

До плюсів віднесемо наступне:

- безсумнівно гнучкіший код;
- можливість повторного використання кожної з трьох складових частин MVC;
- безболісна заміна моделі (інші алгоритми розрахунку, способу зберігання даних і т.д.);
- досить просто перейти від одного подання, до іншого (від HTML до XML або JSON).

Фреймворк Codeignaiter

CodeIgniter – популярний MVC фреймворк з відкритим вихідним кодом, написаний на мові програмування PHP, для розробки повноцінних веб-систем та програм. Розроблено компанією EllisLab (англ.), а також Ріком Еллісом (Rick Ellis) і Полом Бурдіком (Paul Burdick).

Перший публічний реліз фреймворка стався 28 лютого 2006 року. З виходом версії 2.0 28-го січня 2011 CodeIgniter розділився на дві гілки:

- CodeIgniter Core – гілка, яка робить акцент на стабільності, у зв'язку з чим розвивається повільно. Ця версія фреймворку є ядром популярної CMS ExpressionEngine 2.0;
- CodeIgniter Reactor – гілка, яку розвиває співтовариство. Ця гілка розвивається швидко і містить нові можливості.

Особливості:

а) CodeIgniter відрізняє простота, яка досягається завдяки наступним факторам:

1) якісна і повна документація з прикладами, а так само велика спільнота і Wiki;
2) безліч відео-уроків, які можна знайти як на офіційному сайті, так і на сторонніх ресурсах;

б) фреймворк дає свободу програмісту, не створюючи будь-яких структурних обмежень і конвенцій;

в) програмістові не потрібно вчитися користуватися генераторами коду з командного рядка;

г) CodeIgniter працює практично на будь-якому хостинговій платформі, яка має підтримку PHP версії 5.1 і вище;

д) CodeIgniter вважається одним з найшвидших і не вимогливих до ресурсів фреймворків;

е) малий розмір дистрибутива (розмір версії 2.02 складає всього 2.2 Мб);

ж) бібліотеки фреймворка важко або неможливо використовувати в проектах, які не використовують CodeIgniter, тому що вони залежать від компонентів фреймворка;

з) фреймворк дозволяє відмовитися від моделей (незважаючи на те, що це не вітає і не є гарною практикою);

и) документація CodeIgniter пропагує "товсті" контролери та "тонкі" моделі. Валідація та побудова бізнес-логіки відбуваються в основному в контролері. Незважаючи на це, фреймворк дає свободу розробнику, тому він може самостійно вибрати підхід до розробки програми.

До можливостей можна віднести наступні:

- підтримка баз даних MySQL, PostgreSQL, MSSQL, SQLite, Oracle;
- підтримка псевдо-ActiveRecord, який здебільшого повторює синтаксис мови SQL;
- легко розширювана система за рахунок можливості використанн сторонніх і самописних бібліотек, а також доповнення або перевизначення існуючих;
- підтримка як сегментованих так і звичайних URL-ів з передачею параметрів;

- фреймворк містить в собі безліч необхідних бібліотек, які створюють функціонал для роботи з файлами, відправлення електронних листів, валідації форм, підтримки сесій, роботи із зображеннями і так далі;
- володіє можливістю кешування на стороні сервера SQL-запитів і
- генеруються html-сторінок. З версії 2.0 для кешування можуть використовуватися XCache або APC;
- у 2011 році з'явився менеджер пакетів під назвою Sparks, який дозволяє легко встановити сторонні бібліотеки за допомогою командного рядка;
- у 2011 році з'явилося доповнення, яке робить можливим підтримку міграцій;
- підтримка модульності (HMVC) за допомогою додатків не дивлячись на відсутність ORM(*Object-relational mapping*) в стандартом пакеті, існує можливість використання PHP ActiveRecord, Doctrine, Propel та деяких інших ORM після невеликих змін або доповнень у вихідному коді фреймворка;
- за замовчуванням CodeIgniter не використовує шаблонізатори, але є можливість використовувати його власний або один з альтернативних шаблонізаторов (Smarty, TinyButStrong).

Active Record

Вибрати все з таблиці:

```
$this->db->get('table_name');
```

Еквівалент SQL:

```
SELECT * FROM `table_name`
```

Критика фреймворка

Іноді CodeIgniter критикують за незвичайну з точки зору об'єктно-орієнтованого проектування архітектуру. Деякі прихильники CakePHP вважають, що свобода, яка надається програмісту в CodeIgniter, заохочує розхлябаний стиль програмування. І ще:

- моделі в CodeIgniter за замовчуванням не підтримують ORM;
- клас для роботи з базами даних практично неможливо розширювати і змінювати через погано продуману архітектуру;
- не дивлячись на те, що даний фреймворк використовують багато професійних розробників та веб-студії, багато хто вважає, що CodeIgniter підходить тільки новачкам.

Вплив CodeIgniter

CodeIgniter послужив базою для таких фреймворків, як Kohana і Rain Framework, Багато ідей CodeIgniter застосовані в фреймворку Fuel PHP і CodeLighter.

Фреймворк Kohana спочатку був гілкою CodeIgniter і був створений з метою використання всіх можливостей PHP5 зі збереженням простоти і високої продуктивності CodeIgniter. Він відрізнявся від CodeIgniter відсутністю підтримки PHP4 і наявністю деяких додаткових бібліотек (ORM,

Auth та інших). Якщо версії Kohana 2.x все ще чимось були схожі з CodeIgniter, то сучасні версії (3.x) фактично не мають з ним нічого спільного, крім історії.

На CodeIgniter основано безліч CMS: Fuel CMS, MaxSite CMS, Cogear, PyroCMS та інші.

Повну документація по CodeIgniter можна знайти за наступними посиланнями:

- http://code-igniter.ru/;
- http://codeigniter.com/. Бібліотеки, які входять до складу CodeIgniter допомагають вирішити практично всі звичайні завдання. Але крім цього, є два корисні ресурси:

а) codeigniter.com/wiki/ - тут можна знайти безліч корисних рад і додаткових бібліотек;

б) www.ciforge.com/ - цей ресурс спеціально призначений для розміщення проектів, пов'язаних з CodeIgniter. На сьогоднішній день тут розміщено більше двадцяти проектів.

Хід роботи

1. Встановлення та налаштування фреймворка Codeigniter. У мінімальному варіанті для використання CodeIgniter необхідні web-сервер (наприклад, Apache) і інтерпретатор PHP. Крім того, в більшості випадків потрібен сервер баз даних (наприклад, MySQL). Досвідчені програмісти можуть вручну налаштувати ці сервіси. Для початківців підійде пакет Denwer чи IIS, які автоматично виконують необхідні налаштування та встановлюють взаємозв'язки.

Далі будемо розглядати для варіанту з використанням Денвера. Припустимо, що всі необхідні компоненти у вас встановлені. У директорії Денвера, по замовчуванню це Z:/home створюємо папку, наприклад, my_site. Розпаковуємо в цю папку, скачаний з офіційного сайту, архів. Після цього вміст папки my_site матиме приблизно вигляд як на рисунку 8.2.

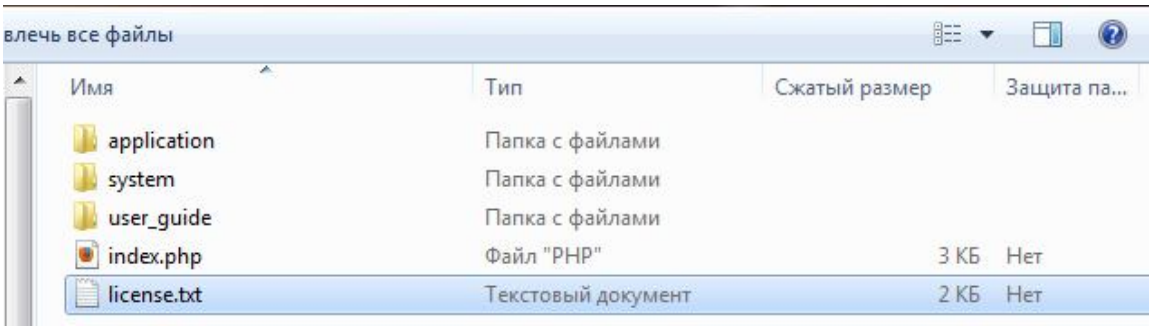

Рисунок 8.2 – Первинний вміст архіву CodeIgniter

Якщо зараз зайти на ваш сайт (http://my_site), то можна побачити сторінку вітання CodeIgniter. В принципі, установка вже завершена, але, по ряду причин, такий варіант далеко не самий кращий.

Головна проблема стосується безпеки. Справа в тому, що дистрибутив CodeIgniter складається з двох частин: папки system зі службовими файлами (в ній, до речі, буде розміщуватися ваш додаток) і основного контролера - файл index.php. У другій папці (user_guide) знаходиться керівництво користувача.

Оскільки папка system знаходиться всередині, то відвідувачі сайту в принципі можуть отримати доступ до файлів, які знаходяться в ній. Саме з цієї причини практично в кожній службової папці CodeIgniter знаходяться файли index.html, які виводять текст «Directory access is forbidden» (Доступ до папки заборонений).

Тому ми дещо зміними розташування папок. Тобто вийде наступна структура, відображена на рисунку 8.3:

Z:/my_site/system/.

Z:/my_site/www/index.php.

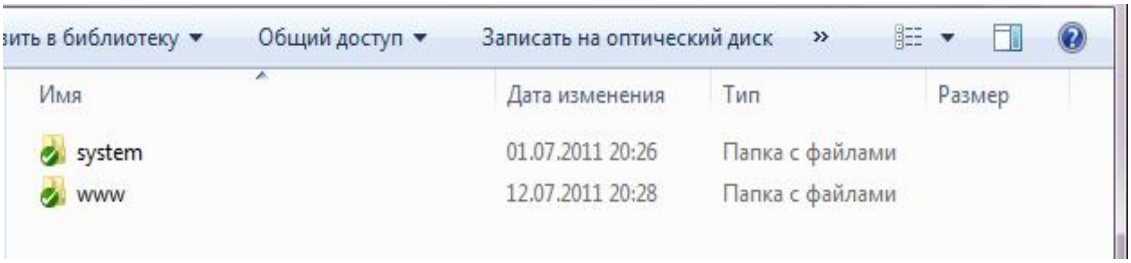

Рисунок 8.3 – Удосконалений вміст архіву CodeIgniter

Тепер для відвідувача доступний тільки index.php. Але наш сайт перестав працювати! Сталося це тому, що index.php не може знайти системні файли. Виправляємо ситуацію.

Відкриваємо index.php, і вказуємо в змінній $system_folder нове розміщення:

$ system_folder = ".. / system";

Якщо є бажання можна видалити файли index.html з усіх системних папок.

Крім того, зверніть увагу на змінну $application_folder (у файлі index.php). У ній зазначено назву папки, в якій знаходиться ваш web-додаток. По-замовчуванню, їй присвоєно значення "application".

Подивимося на структуру цієї папки (system/application). Тут знаходяться:

- config/ - папка з файлами налаштувань сайту;
- controllers/ - папка для контролерів;
- models/ - папка для моделей;
- views/ - папка для уявлень (видів);

- errors/ - тут знаходяться сторінки помилок;
- libraries/ - бібліотеки, які відносяться тільки до вашого додатком;
- hooks/ - хуки – дозволяють змінити поведінку ядра фреймворка (без зміни самих файлів ядра).

Ця структура папок відповідає архітектурі MVC, про яку розповідалось раніше. CodeIgniter містить спеціальні методи, які дозволяють завантажувати потрібні компоненти сайту з цих папок.

Відразу звернемо увагу на обробку помилок. У файлі index.php виконується виклик функції:

error_reporting (E_ALL);

Це означає, що буде виведений звіт про всі помилки, що дуже зручно під час розробки. Але після розміщення готового сайту на сервері, буде краще змінити значення параметра цієї функції. Наприклад, error_reporting (0) повністю відключить виведення повідомлень про помилки.

Далі потрібно здійснити деякі важливі налаштування:

1. Відкрийте файл application/config/config.php в текстовому редакторі і встановіть розташування вашого базового URL. Наприклад:

$config['base_url']= "/".

Задавши значення "/site/" ви автоматично даєте CodeIgniter знати, що там знаходяться файли, з якими він працює, тобто контролери, моделі, види, бібліотеки і т.д. Відповідно і звернення до початкової сторінки буде http://my_site/site.

2. Якщо ви збираєтеся використовувати базу даних, відкрийте файл application/config/database.php в текстовому редакторі і встановіть налаштування бази даних (рисунок 8.4).

В принципі, все готово для написання власного сайту. Ось папки які нам знадобляться:

- system/application/config – файли з настройками фреймворка;
- system/application/controllers – контролери. Докладніше про них шукайте про MVC. Вони відповідають за взаємодію користувача з системою;
- system/application/errors – файли видаються користувачеві при наявності помилок. Їх можна від коректувати;
- system/application/models – моделі. Вони відповідають за вилучення даних з СУБД.
- system/application/views – представлення(вид). Саме їх отримує користувач у свій браузер.

```
1   $active_group = "default";
2   $active_record = TRUE;
3   $db['default']['hostname'] = "localhost";
4   //$db['default']['port'] = 1234;
5   $db['default']['username'] = "root";
6   $db['default']['password'] = "";
7   $db['default']['database'] = "your_db_name";
8   $db['default']['dbdriver'] = "mysql";
9   $db['default']['dbprefix'] = "";
10  $db['default']['pconnect'] = FALSE;
11  $db['default']['db_debug'] = TRUE;
12  $db['default']['cache_on'] = FALSE;
13  $db['default']['cachedir'] = "";
14  $db['default']['char_set'] = "utf8";
15  $db['default']['dbcollat'] = "utf8_general_ci";
```

Рисунок 8.4 – Налаштування бази даних

Починаємо писати власне сторінки нашого сайту, опираючись на патерн MVC та його реалізацію у CodeIgnaiter. Цей код повинен знаходитись в папці controllers і мати назву як у класа тільки в нижньому регістрі, тобто user.php. Щоб створити свій контролер потрібно унаслідуватись від Контроллера CodeIgniter. Будь-який контролер має конструктор, та інші функції в залежності від потреб. У конструкторі зазвичай ініціалізуються поля класу контролера. Також відбувається завантаження моделі, з якою буде спілкуватись контролер, використовуючи вбудовану у фреймворк функцію $this->load->model, яка приймає параметром масив необхідних моделей.

Для прикладу ми взяли контролер, який відповідає за управління користувачами. У ньому є функція registration за допомогою якої користувач зможе зареєструватись на сайті. Відповідно спочатку була створена таблиця з необхідними полями. Як видно контролер не містить якоїсь логіки, а просто передає отримані моделлю данні у вигляд, а якщо при введенні данних, вони проходять валідацію(перевірку), то використовуючи знову ж таки функції моделі, зберігає дані в створену нами таблицю.

В кінці кожної функції повинна біути стрічка $this->renderLayout() оскільки вона відповідає за відображення виду. Назва виду, який буде відображено співпадає з назвою функції, звісно можна вказати інший вид і його розташування, тоді користувач побачить його, а не той, що показався по замовчуванню. Для нашого випадку відобразиться вид /views/user/registration.php, де user назва контролера, а registration.php файл виду(рисунок 8.5).

```
1   class User extends Controller
2   {
3   function __construct()
4   {
5   parent::__construct();
6   $this->load->model(array('/public/user_model', '/public/captcha_model'));
7   $this->data['Title'] = T_('Users');
8   $this->name = T_('Users');
9   }
10  function registration() {
11  $this->data['Title'] = T_('Register');
12  $this->validation->set_fields($this->user_model->fields);
13  $rules = array('first_name', 'last_name', 'email', 'email_confirm', 'password',
14  'password_confirm', 'payment_method', 'captcha_code');
15  $this->validation->set_rules($this->user_model->getRules($rules));
16  $this->validation->set_message('__captcha', T_('Incorrect verification code.'));
17  $this->validation->set_message('__correct_pass', T_("Password lenght must consist to
18  both letters and numbers with at least 1 capital letter"));
19  if ($this->input->post('register')){
20  if ($this->validation->run() === TRUE){
21  $user_first_name   = $this->validation->first_name;
22  $user_last_name    = $this->validation->last_name;
23  $user_email   = $this->validation->email;
24  $user_password  = $this->validation->password;
25  $user_status  = 3;
26  $arr['user_first_name'] = $user_first_name;
27  $arr['user_last_name'] = $user_last_name;
28  $arr['user_email'] = $user_email;
29  $arr['user_password'] = $user_password;
30  $arr['payment_method'] = 1;
31  $this->user_model->insert($arr);
32  $this->view ='/user/verification_email';
33  }else{
34  $this->view ='/user/registration';
35  }
36  }
37  $this->renderLayout();
38  }
39  }
```

Рисунок 8.5 – Приклад написання сторінки сайту і створення контролера

Модель містить назву таблиці з якою вона працює, а також назви полів та правила валідації (required, наприклад, означає, що при заповненні форми це поле не може залишитись пустим, тобто є необхідним для заповнення). Валідація може визначати вміст поля, кількість символів і таке інше, а також ссилатись на функцію, яка буде перевіряти правильність введення (callback). Модель розташовується у папці models у файлі user_model.php (рисунок 8.6).

```
1   class User_model extends Model
2   {
3   public $table = 'users';
4   function __construct()
5   {
6   parent::__construct();
7   $this->fields = array(
8   'full_name'              => T_('Full Name'),
9   'friendemail'            => T_('Friend Email'),
10  'id'                     => '',
11  'first_name'             => T_('First Name'),
12  'last_name'              => T_('Last Name'),
13  'email'                  => T_('E-Mail'),
14  'email_confirm'          => T_('Confirm E-Mail'),
15  'password'               => T_('Password'),
16  'new_password'           => T_('Password'),
17  'password_confirm' => T_('Confirm Password'));
18  $this->rules = array(
19  'first_name'             => "trim|required|max_length[255]",
20  'last_name'              => "trim|required|max_length[255]",
21  'email'              => "trim|required|valid_email|max_length[64]|callback___loginFree",
22  'friendemail'      => "trim|required|valid_email|max_length[64]",
23  'email_confirm'  =>"trim|matches[email_confirm]|required|valid_email|max_length[64]",
24  'password'=> "trim|required|matches[password_confirm]|min_length[5]
25  |max_length[64]|callback___correct_pass",
26  'new_password'  => "trim|[password_confirm]|min_length[8]|max_length[64]|
27  callback___correct_pass",
28  'password_confirm' => "trim|min_length[8]|max_length[64]");
29  }
30  function insert($data)
31  {
32  if (!empty($data))
33  {
34  if (isset($data['user_password']))
35  {$data['user_password']= $this->__cryptPassword($data['user_password']); }
38  $data['user_created'] = date('Y-m-d H:i:s');
39  parent::insert($data);
40   if ($this->db->insert_id() > 0)
41  {
42  $this->log->writeMessage('done', T_('User was successfully created.'));
43  return TRUE;
44  }
45  else
46  {
47  $this->log->writeMessage('error', T_('Error! User was not created.'));
48  return FALSE;
49  }}}
50  function __cryptPassword($password = null) {
51  return md5($password . SALT); }}
```

Рисунок 8.6 – Приклад створення моделі

Останнім елементом є вид, який має назву функції контролера, та розташовується в папці views.

Опускаємо стандартні теги і розглянемо тільки контент сторінки, який буде розташовуватись безпосередньо в тегу body. Як бачимо у відповідні поля виводяться дані, що записані в масив data у контролері (рисунок 8.7).

```
1   <div id="content">
2   <div id="rgLeftContent">
3   <div id="rgTitle" class="titleI">You do not pay a fee to Register!</div>
4   <div id="rgLeftBottom" class="blockstyle"></div>
5   </div>
6   <div id="rgCenterContent">
7   <div id="rgTitle" class="titleI"><?php echo $Title; ?></div>
8   <div id="rgForm" class="blockstyle form">
9   <?php echo form_open($this->curl . '/registration', 'id="myform" name="myform"
10  autocomplete="off"')?>
11  <?php echo $this->validation->error_string?>
12  <div id="rgFirstName" class="formElement">
13  <div class="blue formText"><?php echo $this->user_model->fields['first_name'];
14  ?></div>
15  <?php echo form_input('first_name', $this->validation->first_name,
16  'class="formInput"')?>
17  </div>
18  <div id="rgLastName" class="formElement">
19  <div class="blue formText"><?php echo $this->user_model->fields['last_name'];
20  ?></div>
21  <?php echo form_input('last_name', $this->validation->last_name,
22  'class="formInput"')?>
23   </div>
24  <div id="rgNickName" class="formElement">
25  <div class="blue formText"><?php echo $this->user_model->fields['nickname'];
26  ?></div>
27  <?php echo form_input('nickname', $this->validation->nickname, 'class="formInput"')?>
28  </div>
29  <div id="rgEmail" class="formElement">
30  <div class="blue formText"><?php echo $this->user_model->fields['email']; ?></div>
31  <?php echo form_input('email', $this->validation->email,
32  'class="formInput"')?>
33  </div>
34  <div id="rgConfirmEmail" class="formElement">
35  <div class="blue formText"><?php echo $this->user_model->fields['email_confirm'];
36  ?></div>
37  <?php echo form_input('email_confirm', $this->validation->email_confirm,
38  'class="formInput"')?>
39  </div>
40  <div id="rgEmailAbout" class="formElement">
```

Рисунок 8.7 – Приклад створення стандартних тегів та розташування контент сторінки, аркуш - 1

```
41  <div class="lightred justify"><?php echo T_('You will use your e-mail to login to HN so
42  we must make sure there are no typos; same reason for confirming password.')?></div>
43  </div>
44  <div id="password" class="formElement">
45  <div class="blue formText"><?php echo $this->user_model->fields['password']; ?>
46  </div>
47  <?php echo form_password('password', $this->validation->password,
48   'class="formInput"')?>
49  </div>
50  <div id="confirmPassword" class="formElement">
51  <div class="blue formText"><?php echo $this->user_model->fields
52  ['password_confirm']; ?></div>
53  <?php echo form_password('password_confirm', $this->validation->password_confirm,
54  'class="formInput"')?>
55  </div>
56  <div class="formElement" id="captcha">
57  <img class="captcha" src="<?= rtrim(site_url(''), '/').'/captcha/image/'.set_value
58  ('captcha_id', $this->session->userdata('session_id'));?>" alt="captcha"  style =
59  " margin :10px "/><br />
60  <input name="captcha_id" value="<?= set_value('captcha_id', set_value('captcha_id',
61  $this->session->userdata('session_id')));?>" type="hidden" />
62  <?php echo form_input('captcha_code', '', 'class="input-text formInput"
63  style="width: auto;"')?>
64  </div>
65  <input id="rgContinue" class="red buttonStyle" type="submit" value="
66  <?php echo T_('CONTINUE'); ?>" name="register" />
67  <?php echo form_close()?>
68  </div>
69  </div>
70  <div style="float:left; height:540px; position: relative; margin: 0 0 8px; width: 106px;">
71  <div id="emblem"></div>
72  </div>
73  </div>
```

Рисунок 8.7 – Приклад створення стандартних тегів та розташування контент сторінки, аркуш - 2

За аналогічним принципом створюються інші сторінки сайту і реалізуються переходи між ними. Звичайно можна зробити сайт на одному контролері, але як показує практика для великих проектів це буде дуже непродуктивне рішення, оскільки міститиме такий контролер багато стрічок коду, що зменшує як ефективність його редагування, так і збільшує час завантаження сторінки, який витрачається на функції контролера, що не будуть використовуватися. Отож бажано логічно розподілити сайт на сукупність контролерів, моделей і відображень, кожен з яких буде використовуватись там де це необхідно.

І на завершення, при написанні сайту варто добре розібратись з архітектурою майбутнього проекту, це збереже дорогоцінний час у майбутньому.

А спосіб вибирайте який вам зручніший, чи це готовий каркас, чи фреймморк, чи «чистий» php. Це залежить особисто від кожного проекту і прийняття правильного рішення у подальшому усуне виникнення небажаних проблем.

2. Розробка WEB-сайту з використанням CMS WordPress

Після встановлення Denwer на свій комп'ютер проведемо дослідження однієї з CMS. Обираємо CMS WordPress. WordPress написаний на PHP і для його роботи потрібна база даних MySQL. Це досить легкий в розумінні движок, але дуже функціональний і гнучкий. Давайте його і встановимо.

Встановлення WordPress на Denwer

Щоб встановити WordPress на Денвер, виконайте наступні кроки:

1) Завантажте архів з WordPress. Всі останні версії вордпресс тут. Распакуйте архів.

2) Тепер створимо віртуальний хост. Для цього треба зайти в C: \ WebServers \ home або в Z: \ home (все залежить від того, які папку і букву диска ви вибрали в процесі установки) і створюємо в ній папку. Ім'я папки задайте будь-яке за бажанням, але латинськими літерами. Задамо ім'я cms1.

У створеній папці створіть папку www. Після створення у мене вийшов такий шлях: Z: \ home \ cms1 \ www.

3) Скопіюйте всі файли з папки wordpress, яка перебувала в архіві, і вставте їх в папку Z: \ home \ cms1 \ www.

4) Перезапустіть Denwer, клікнувши по ярлику Restart Denwer.

5) Відкрийте браузер і введіть localhost (рисунок 8.8).

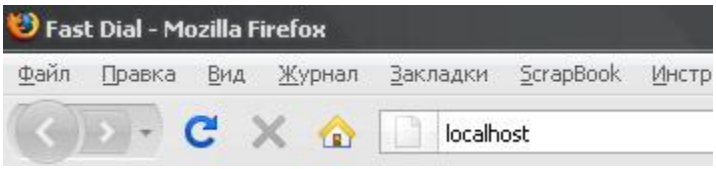

Рисунок 8.8 – Запуск домашньої сторінки

Якщо ви бачите сторінку з написом «Ура, запрацювало!», то все зроблено правильно і Денвер працює.

6) Тепер нам потрібно створити базу даних (БД) для WordPress. Для роботи з БД використовується phpMyAdmin, яка вже встановлена в Денвері. Щоб увійти до phpMyAdmin, спустіться вниз сторінки, яка з'явилася на 5 пункті (localhost) і клацніть на посилання: http://localhost/Tools/phpMyAdmin Проверка MySQL и phpMyAdmin (Або в адресной строці браузера введіть http://localhost/Tools/phpMyAdmin).

Щоб працювати з базами даних потрібно створити користувача MySQL з паролем. За умовчанням в phpMyAdmin в Денвері створено користувач root без пароля. Деякі CMS вимагають щоб у користувача MySQL був па-

роль. Тому можна відредагувати вже створеного користувача або створити нового. Користувач створюється один раз і потім його можна використовувати для установки інших скриптів або CMS, які працюють з MySQL.

Щоб створити користувача, клацніть на посилання «Привилегии».

Потім зайдіть по посиланню «Добавить нового пользователя». Після цього з'явиться вікно створення нового користувача (рисунок 8.9).

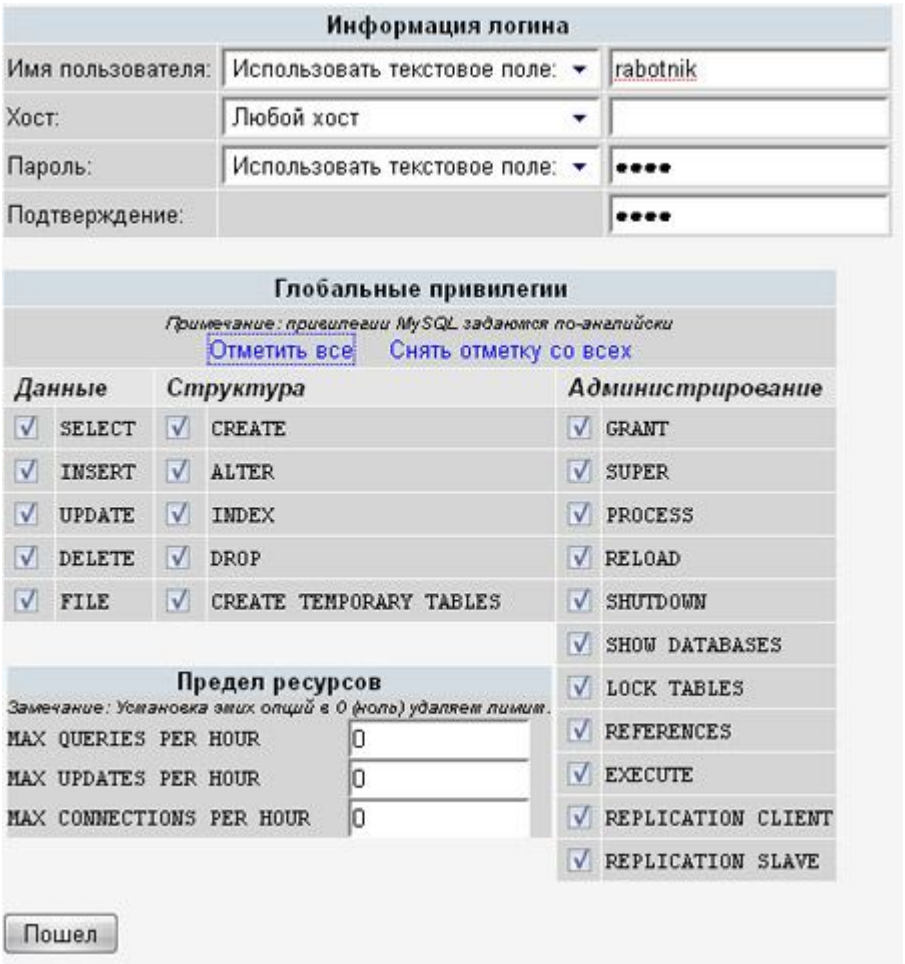

Рисунок 8.9 – Вигляд вікна створення нового користувача

Введіть Ім'я користувача, пароль (в 2-х полях) і відзначте всі привілеї (посилання «Отметить все»), натисніть кнопку «Пошел». Добре запам'ятайте пароль та ім'я, вони завжди знадобляться.

Користувач створений і тепер потрібно створити базу даних MySQL для WordPress.

Щоб створити базу поверніться на головну сторінку phpMyAdmin (http://localhost/Tools/phpMyAdmin/) і там, де «Создать новую БД» введіть латинськими літерами ім'я вашої бази даних і натисніть кнопку «Создать» (рисунок 8.10).

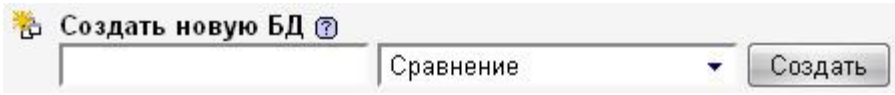

Рисунок 8.10 – Створення нової бази даних

Ім’я можна вибрати будь-яке. Вводимо wp.

БД wp була створена.

7) Віртуальний хост з файлами WordPress і база даних готові. Тепер приступаємо до встановлення CMS.

Введіть у рядку браузера ім’я вашого віртуального хоста, тобто назва вашої папки. У нас це cms1.

Після цього завантажиться вікно в якому буде написано:

Можливо немає файлу wp-config.php і т.д

Натисніть на кнопку «Создать конфигурационный файл». Після чого, завантажитися нове вікно, в якому потрібно натиснути кнопку «Поехали».

Останній крок установки. Тут вам потрібно ввести (рисунок 8.11): ім’я бази даних, ім’я користувача бази даних і пароль користувача бази даних (ті, які ви створювали у п. 6).

Рисунок 8.11 – Визначення параметрів для підключення до бази даних

Потім натисніть кнопку «Запуск инсталляции«. У наступному вікні вводите назву блогу і e-mail. Потім система видасть вам логін admin і пароль. Запишіть пароль. Він потрібен для входу в адмін панель WordPress.

Також БД можна створити за допомогою консолі за допомогою команди (рисунок 8.12): Create database WordPress;

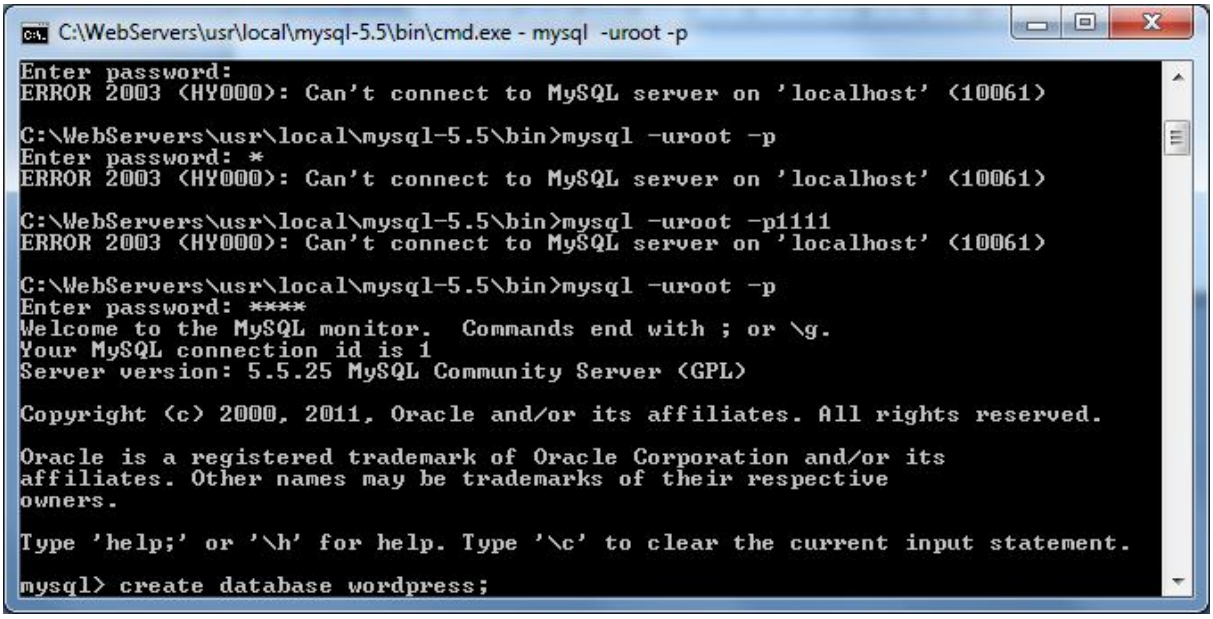

Рисунок 8.12 – Командний спосіб створення бази даних

Далі перейдемо за адресою localhost і побачимо як виглядає наш сайт. Для входу в адміністративну панель потрібно перейти на localhost/wp-admin, ввести логін та пароль які ввели при реєстрації (рисунок 8.13).

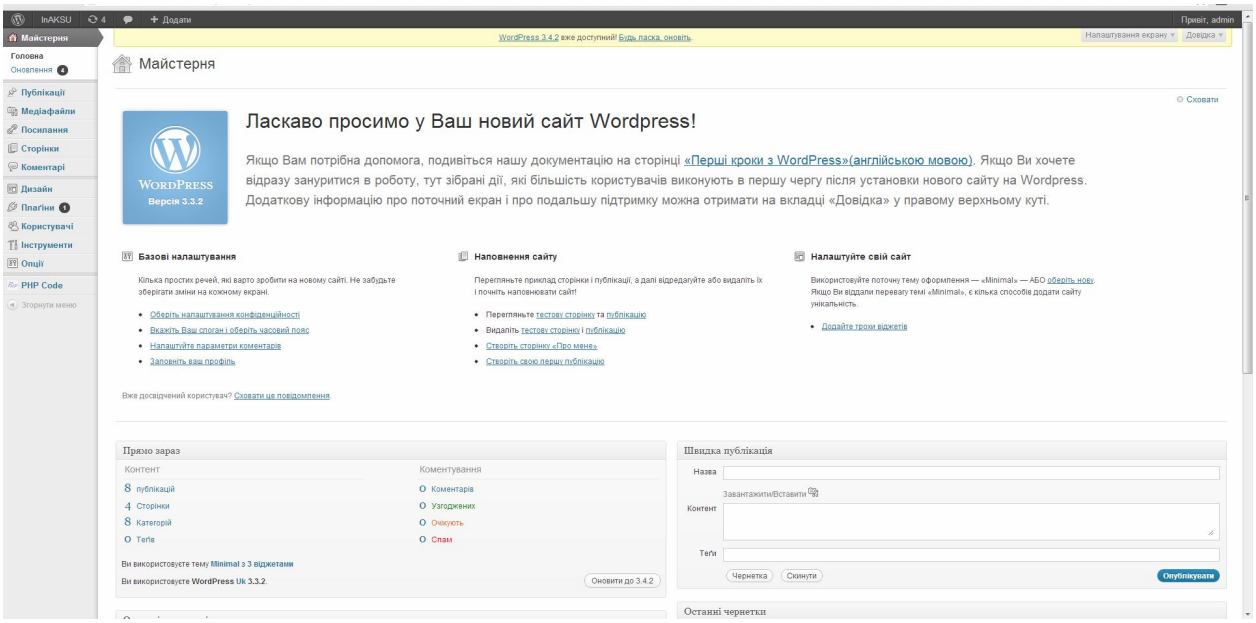

Рисунок 8.13 – Вигляд вікна адміністрування

Зміна шаблону. Для зміни теми потрібно перейти в меню дизайн-> теми, в цьому пункті можна обрати будь яку тему. Після вибору теми, встановимо її. Ми обрали тему minimal. В меню Minimal theme option (рисунок 8.14) можна змінювати деякі параметри теми, але не всі, це полегшує ро-

боту початківцям, але для більш тонкого налаштування потрібно редагувати сам шаблон теми.

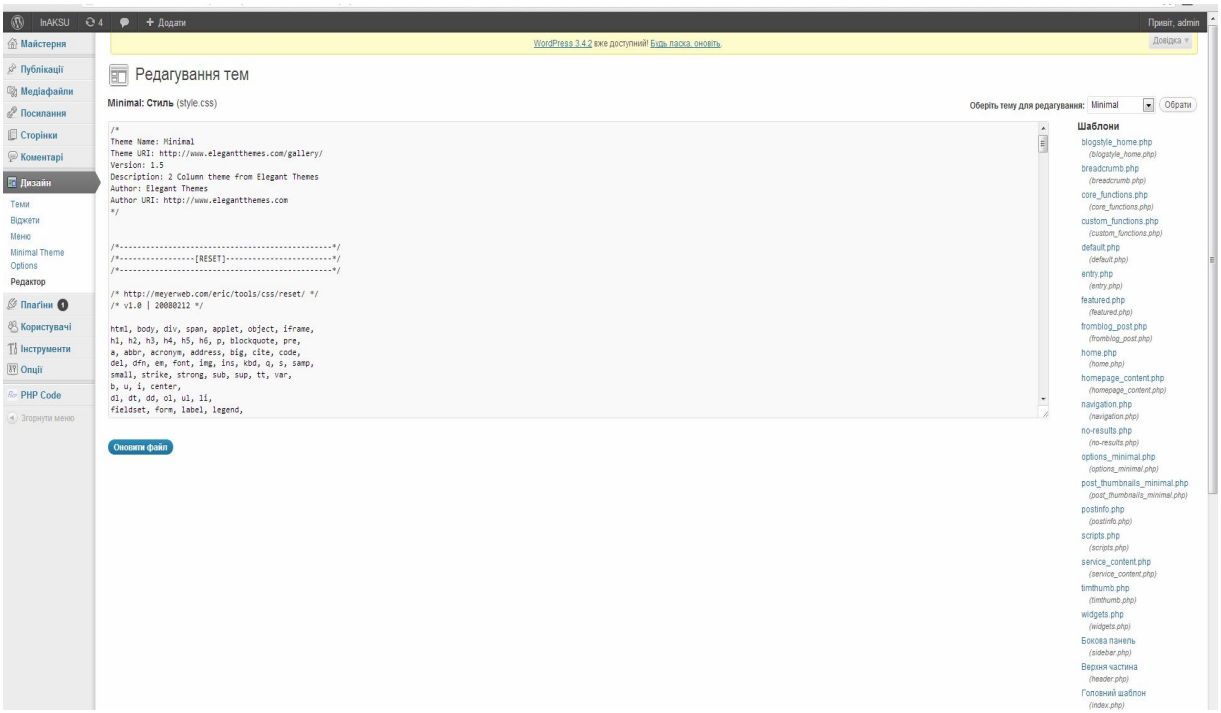

Рисунок 8.14 – Вигляд вікна для визначення параметрів теми

У меню праворуч розташовані усі файли теми, файли php та css в файлі header.php. Ми змінили вигляд меню.

Далі нам знадобиться розширення PHP CODE для можливості написання php коду в статях.

Напишемо невеликий скрипт для виведення статей кожного автора (рисунок 8.15).

По завершенню нашої роботи, отримаємо готовий продукт, який зображено на таких рисунках:

- Рисунок 8.16 – вкладка «Викладачі».
- Рисунок 8.17 – вкладка «Про інститут».
- Рисунок 8.18 – вкладка «Дисципліни».
- Рисунок 8.19 – вкладка «Архів».

```
1   <?php
2   $sql="select * from wp_usermeta";
3   $result=mysql_query($sql);
4   $i=0;
5   while($row=mysql_fetch_array($result))
6   { if($row['meta_key']=='first_name' ||$row['meta_key']=='last_name' ||
7   $row['meta_key']=='nickname'){
8   $i++;
9   if($row['meta_key']=='first_name' ||$row['meta_key']=='last_name' ){
10  echo $row['meta_value'];
11  echo "  ";
12  }
13  if($row['meta_key']=='nickname'){
14  $temp="http://testtestttt.16mb.com/wordpress/archives/author/".$row['meta_value'];
15  echo "<A href='$temp'>  Перегляд статтей автора</A>";}
16  }
17  if($row['umeta_id']%16==0){echo "<br><br>";}
18  }
19  ?>
```

Рисунок 8.15 – Вміст скрипту для виведення статей кожного автора

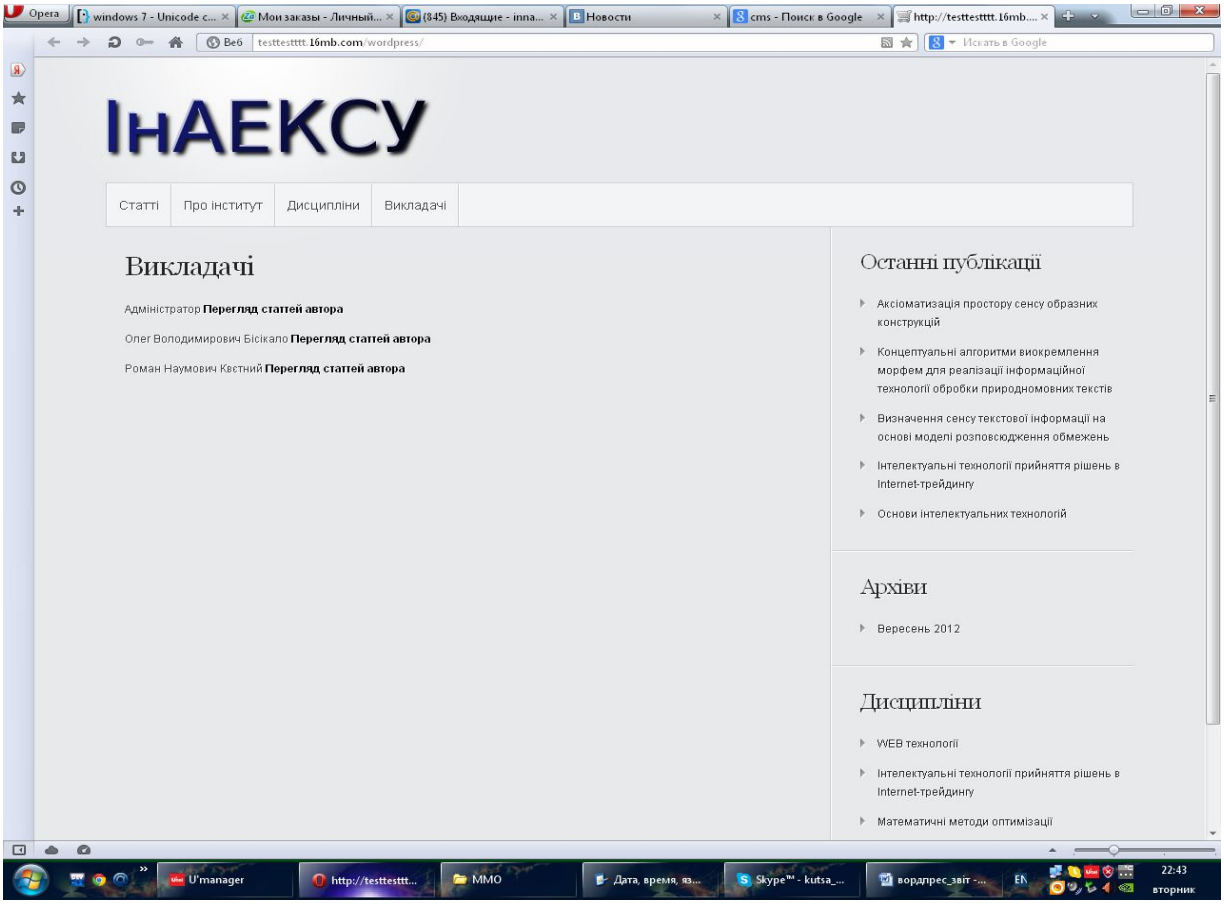

Рисунок 8.16 – Вигляд вкладки «Викладачі»

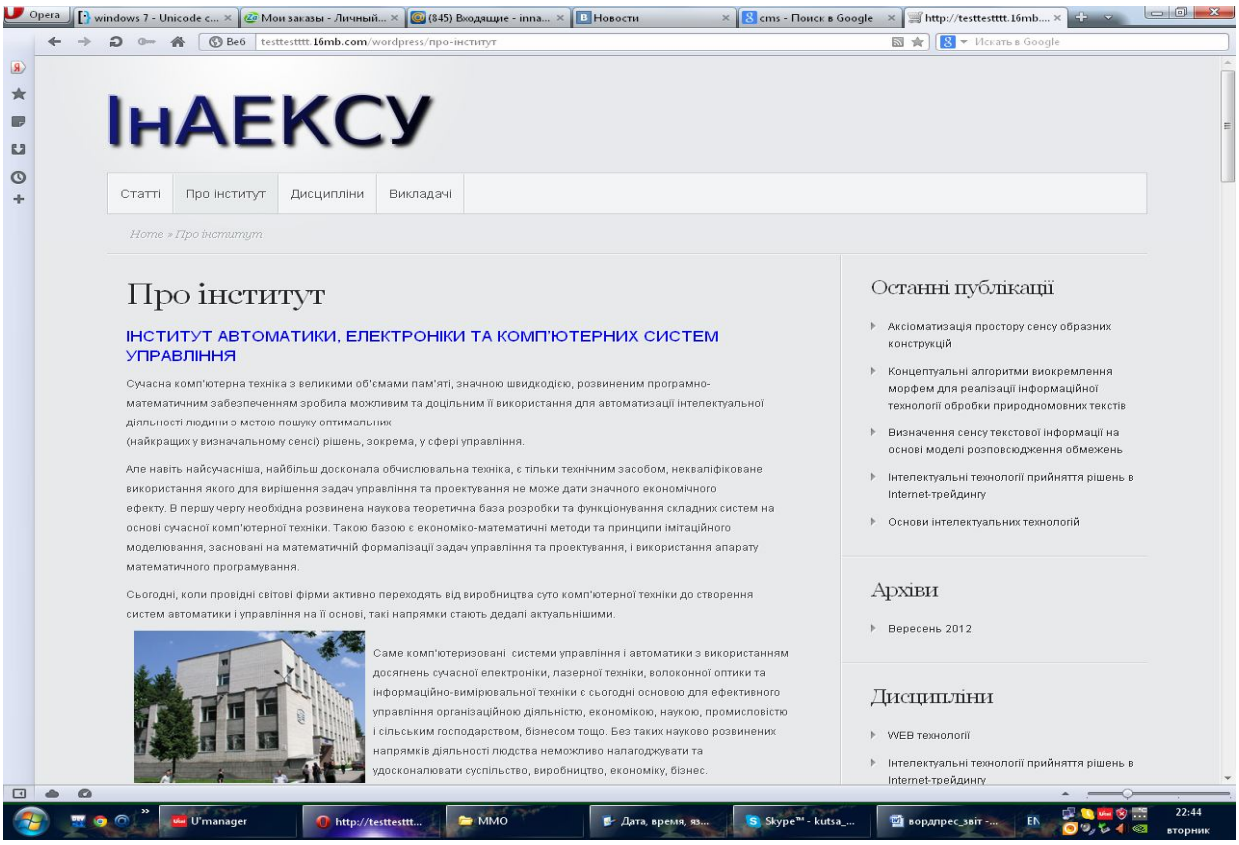

Рисунок 8.17 – Вигляд вкладки «Про інститут»

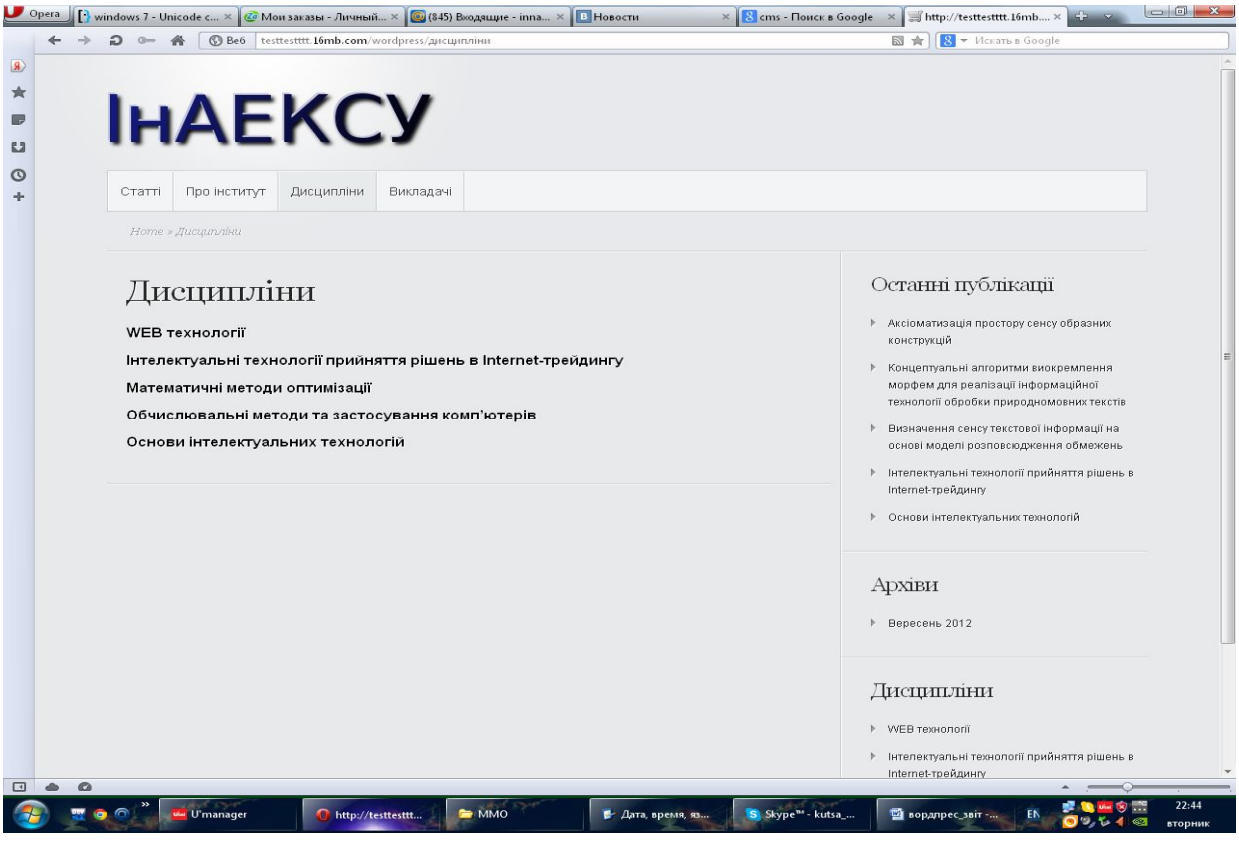

Рисунок 8.18 – Вигляд вкладки «Дисципліни»

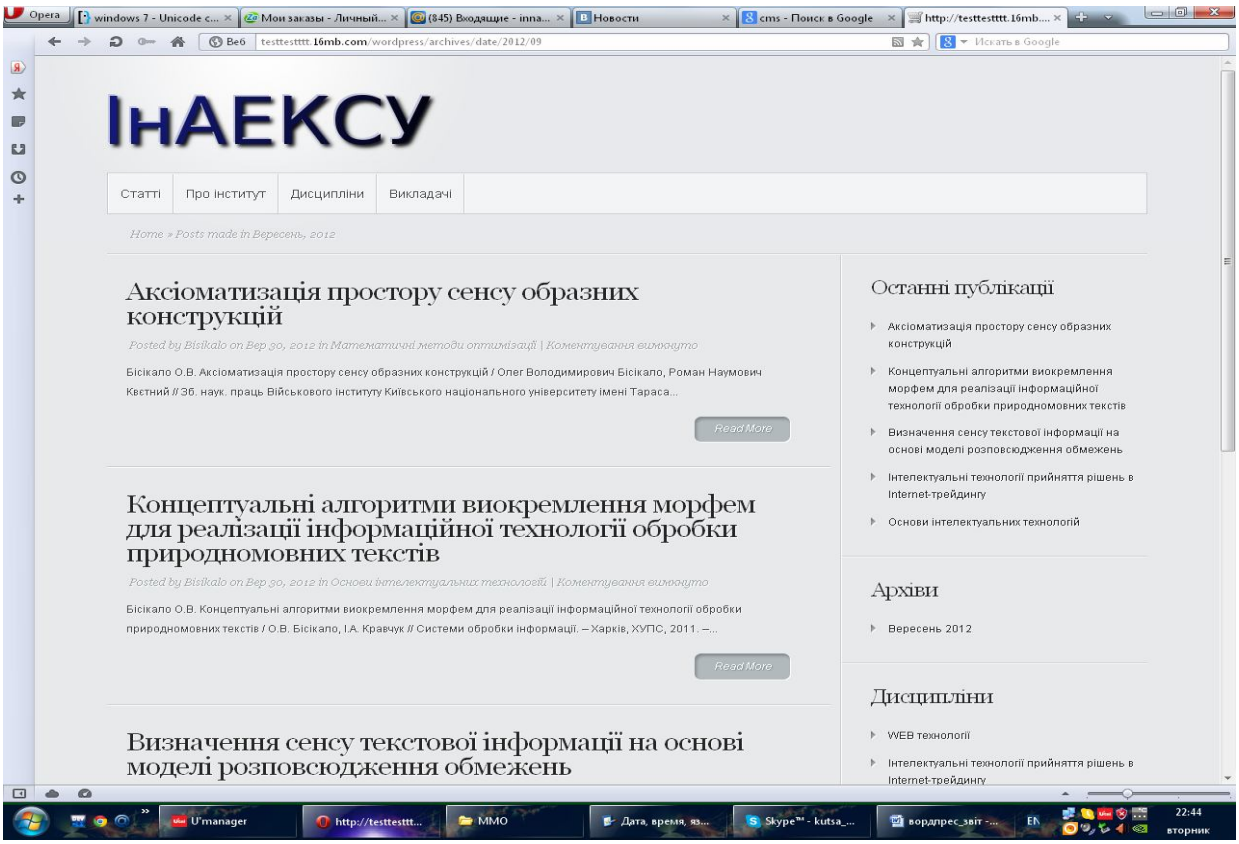

Рисунок 8.19 – Вигляд вкладки «Архів»

Отже, ми отримали інформаційний продукт - ВЕБ-додаток на основі CMS WordPress.

Вміст звіту:

а) тема і мета лабораторної роботи;
б) у відповідності до кроків ходу роботи навести:
   1) графічне представлення інженерних дій у вигляді схем програм, схем взаємодії програмних модулів, діаграм використання або інших UML-діаграм;
   2) лістинг програмних модулів;
   3) результати виконання програмних модулів;
в) змістовні висновки за результатами роботи.

Питання для самоконтролю:

1. Що таке CMS?
2. Які CMS вам відомі?
3. Що таке фреймворк?
4. Які веб-фреймворки вам відомі?
5. Що таке схема Модель – Вид – Контроллер?

6. Як історично створювалася концепція MVC?
7. Яке призначення має концепція MVC?
8. У чому полягає сутність концепції MVC?
9. Які шаблони проектування застосовуються у MVC?
10. Які переваги та недоліки притаманні MVC-системам?
11. Що таке CodeIgniter?
12. Чим CodeIgniter відрізняється від своїх аналогів?
13. Які переваги та недоліки має CodeIgniter**?**
14. Який вплив мав та має має CodeIgniter на інші фреймворки**?**
15. Охарактеризуйте CMS WordPress?
16. Як встановити WordPress?
17. Як створити користувача MySQL з паролем?
18. Як створити базу даних для WordPress?
19. Як змінювати шаблон у CMS?
20. Що можна вважати інформаційним продуктом для технології на основі CMS?

РЕКОМЕНДОВАНА ЛІТЕРАТУРА

1. Хольцнер С. PHP в примерах / Стивен Холцнер; пер. с англ. – М.: ООО «Бином-пресс», 2007. – 352 с.
2. Ульман Л. Основы программирования на PHP / Л. Ульман; пер. с англ. – М.: ДМК Пресс, 2001. – 288 с.
3. Колисниченко Д. Н. Самоучитель PHP 5.0 / Д. Н. Колисниченко – СПб.: Наука и техника, 2004. – 566 с.
4. Птицын К. А. Серверы Linux: самоучитель / К. А. Птицын. – Киев: Диалектика, 2003. – 208 с.
5. Матвієнко О. В. Internet-технології: проектування Web-сторінки: навч. посіб. / О. В. Матвієнко, І. Л. Бородіна – К.: Центр навчальної літератури, 2004. – 154 с.
6. Бенкен Е. С. PHP, MySQL, XML: программирование для Интернета + CD / Е. С. Бенкет – СПб.: BHV-Санкт-Петербург, 2008. – 352 с.
7. Шапошников И. В. PHP 5.1. Учебный курс / И. В. Шапошников – СПб.: Питер, 2007. – 192 с.
8. Леки-Томпсон Э**.** PHP 5 для профессионалов / Э**.** Леки-Томпсон – М.: Вильямс, 2006. – 608 с.
9. Зольников Д. С. PHP 5 / Д. С. Зольников – М.: НТ Пресс, 2007. – 256 с.
10. Ленгсторф Д. PHP и jQuery для профессионалов = Pro PHP and jQuery / Джейсон Ленгсторф. – М.: «Вильямс», 2010. – с. 352.
11. Валейд Д. PHP 5 для "чайников": Пер. с англ. / Джордж Валейд – М.: Диалектика-Вильямс, 2005. – 320 с.
12. Інформаційні матеріали по PHP [Електронний ресурс] – Режим доступа: http://www.master-web.com.ua/book/16 – Назва з екрану.
13. PHP: Hypertext Preprocessor [Електронний ресурс] – Режим доступа: http://www.php.net/ – Назва з екрану.
14. MySQL: The world's most popular open source database [Електронний ресурс] – Режим доступа: http://www.mysql.com/ – Назва з екрану.
15. Programmer's Klondike: Веб технології, PHP [Електронний ресурс] – Режим доступа: http://www.programer.org.ua/index.php?part=9&mainpart=2&filt=1&page=1 – Назва з екрану.
16. Современный учебник JavaScript [Електронний ресурс] – Режим доступа: http://learn.javascript.ru/ – Назва з екрану.
17. В глубь языка Python: Для программистов. [Електронний ресурс] – Режим доступа: http://ru.diveintopython.net/odbchelper_divein.html#odbchelper.divein – Назва з екрану.

Додаток А (довідковий). Перелік основних безкоштовних хостингів та послуг, які вони надають

Таблиця А.1 – Перелік основних безкоштовних хостингів та послуг

Адреса	Опис послуг
1	2
Hostinger.com.ua	2Gb дискового простору; 100Gb трафіку; без реклами; підтримка PHP/MySQL; простий онлайн-сайт конструктор; панель управління cPanel; дозволена ваша власна реклама; FTP доступ; підтримка Zend Optimizer, Ioncube, Curl, GD2 lib, XML; необмежена кількість доменів; поштові сервіси POP3, Webmail , IMAP; Веб файл менеджер; phpMyAdmin. Мінуси: заборона розміщувати відео з YouTube(видаляють сайт); перебої з FTP доступом;
Iplot.ru	Тут Ви можете безкоштовно розмістити свій сайт реєстрація автоматична, доступ - моментально. Ви отримаєте 50 мегабайт дискового простору, безкоштовне доменне ім'я виду: ваш_логін.iplot.ru Плюс: CGI, MySQL, SSI, Perl, PHP, доступ по FTP. Без реклами! Мінуси: нестабільність, низька швидкість
KM.ru	70 Mb дискового простору. Плюс: PHP, Perl, MySQL і т.д. і зручність у використанні. Мінус: приймають далеко не всіх, швидкість частенько залишає бажати кращого. Банер на головній сторінці і pop up.
BY.ru	Необмежений дисковий простір. Плюс: PHP, Perl, MySQL (1 таблиця) і т.д. і готові скрипти. Мінус: реклама pop up, що відкривається ще до завантаження сторінки і маса обмежень призначених для того, щоб ви купили у них платний хостинг.
TakeInet.Ru	Плюс: CGI, PHP, SSI, MySQL, доступ по FTP. Розміщення банерів в скриптах не обов'язково. Домен типу your_name.takeinet.ru або your_name.iwebs.ru. Встановлені скрипти: Гостьові книги, Форуми, Голосування, Лічильники, Аукціони, Чати і т.д. Безкоштовна пошта з Pop3-доступом. Допомога початківцям у створенні і розкрутці сайту. Конструктор сайтів дозволить створити сайт без знання HTML.
Traktir.ru	Дисковий простір необмежена (хоча за замовчуванням надають 15 Mb). Домен типу your_name.traktir.ru або your_name.traktir.org. За запитом можна реєструвати домени четвертого рівня. Плюс: підтримка SSI, PHP3, PHP4, CGI, листів розсилки доступу по FTP. За запитом надаються доступ до баз даних MySQL та поштові скриньки. Хостинг надається тільки для цікавих некомерційних проектів або особистих сторінок, зроблених з душею.

Продовження таблиці А.1.

1	2
fh.com.ua	Місце до 150 Мб (можна і більше), безкоштовний хостинг для готових проектів. При відвідуваності більше 100 хостів на день Ви отримаєте: Доменне ім'я your_name.com.ua (*. Kiev.ua, *. Org.ua ...), Місце до 50 Мб (можна і більше), Доступ по FTP, Можливість виконувати свої CGI, Perl скрипти, PHP, SSI, Поштова скринька (POP3, або форвард). При відвідуваності більше 500 хостів на день Ви отримаєте Доменне ім'я your_name.com (*. Net, *. Org) і поштову скриньку (POP3, або форвард) до свого домену. Натомість Ви повинні будете на всіх сторінках свого сайту розміщувати або один банер 468 * 60, або два банери 100 * 100.
Hoha.ru	Доменне ім'я третього рівня (ваш_домен.hoha.ru або ваш_домен.fatal.ru); 50 Мб дискового простору під Ваш сайт; Поштова скринька ваш_домен@hoha.ru або ваш_домен@fatal.ru, 3 додаткових поштових скриньок виду ваше_імя@ваш_домен.hoha . ru або, відповідно,ваше_імя@ваш_домен.fatal.ru. Плюс: Підтримка CGI, PHP, SSI; FTP доступ до вашого сайту.
Narod.ru	100 Мб дискового простору, згодом дисковий простір може бути автоматично збільшена до будь-якого обсягу. Плюс: Безкоштовний форумчат, пошуковик по Вашим ресурсів. Мінус: відсутність підтримки скриптів.
Iwebland.com	100 Мб дискового простору, згодом дисковий простір може бути автоматично збільшена до будь-якого обсягу. Плюс: Безкоштовний форумчат, пошуковик по Вашим ресурсів. Мінус: відсутність підтримки скриптів.
Iwebland.com	Єдиний зарубіжний хостер в даному списку. Я спеціально не акцентував вашу увагу на іноземних хостерах, так як і вітчизняних вистачає. Плюс iwebland.com: PHP, MySQL, непогана швидкість. Мінуси: нестабільність, реклама у вигляді пошукової сторінки, яка одноразово з'являється в новому вікні браузера на задньому плані при сходженні з домену.
WebServis.ru	100 Mb дискового простору, 1 поштова скринька, домен третього рівня типу: ваше_імя.webservis.ru, ваше_імя.al.ru, ваше_імя.vipcentr.ru, ваше_імя.vippochta.ru, ваше_імя.viptop.ru, ваше_імя.vipshop.ru, ваше_імя.bos.ru, ваше_імя.bip.ru, ваше_імя.dem.ru, ваше_імя.far.ru, ваше_імя.fud.ru, ваше_імя.hobi.ru, ваше_імя.hop.ru, ваше_імя.r2.ru, ваше_ім'я. gok.ru, ваше_імя.pp.ru! Плюс: підтримка CGI, SSI, PHP, доступ по FTP. Є готові форуми, голосування, новинні стрічки, лічильники багато іншого.

Продовження таблиці А.1.

1	2
Hut.ru	30 Mb дискового простору. Однак за наявності у Вашого сайту достатньо великої відвідуваності, Вам можуть видати додатковий дисковий простір. Після реєстрації Ви отримуєте легко запам'ятовується адресу: ваше_імя.hut.ru. Плюс: PHP, CGI, MySQL, FTP-доступ та shell-аккаунт і багато іншого. Мінус: Після завантаження сторінки, зверху автоматично вставляється банер 468x60.
RDCOM.ru	необмежений дисковий простір, ваше_імя.rdcom.ru. Плюс: підтримка CGI, PHP, SSI, MySQL. FTP-доступ. Мінус: реклама pop up, низька швидкість.
Wallst.ru & Webservis.ru	необмежений дисковий простір, your_name.al.ru, your_name.webservis.ru, your_name.vipcentr.ru, your_name.vippochta .ru, your_name.bos.ru, your_name.viptop.ru, your_name.vipshop.ru. Плюс: підтримка CGI, PHP. FTP-доступ. Має передвстановлені скрипти з можливістю повної зміни дизайну на ваш розсуд (Гостьові книги, Форуми, Чати, емаіл форми, Служби знайомств, Пошук по серверу, емеіл підписка / розсилка, Листівки, Голосування). Мінус: рекламний банер у верхній частині.
Igg.ru	30 Mb дискового простору. Плюс: підтримка CGI, PHP, MySQL, SSI. Мінус: mp3 не дозволялось, сторінки досить жорстко модеруються.

Навчальне видання

Веб-технології
Лабораторний практикум

Редактор В. Дружиніна

Укладачі: Олег Володимирович Бісікало
Роман Васильович Маслій
Євген Анатолійович Паламарчук
Рима Олександрівна Яцковська

Оригінал-макет підготовлено Р. Маслій

Підписано до друку ……..
Формат 29,7×42¼. Папір офсетний.
Гарнітура Times New Roman.
Друк різографічний. Ум. друк. арк. …...
Наклад … пр. Зам. №

Вінницький національний технічний університет,
навчально-методичний відділ ВНТУ.
21021, м. Вінниця, Хмельницьке шосе, 95,
ВНТУ, к. 2201.
Тел. (0432) 59-87-36.
Свідоцтво суб'єкта видавничої справи
серія ДК № 3516 від 01.07.2009 р.

Віддруковано у Вінницькому національному технічному університеті
в комп'ютерному інформаційно-видавничому центрі.
21021, м. Вінниця, Хмельницьке шосе, 95,
ВНТУ, ГНК, к. 114.
Тел. (0432) 59-85-32,
publish.vntu.edu.ua; email: kivc.vntu@gmail.com.
Свідоцтво суб'єкта видавничої справи
серія ДК № 3516 від 01.07.2009 р.