

**ТЕХНІЧНИЙ КОЛЕДЖ ТЕРНОПІЛЬСЬКОГО НАЦІОНАЛЬНОГО
ТЕХНІЧНОГО УНІВЕРСИТЕТУ ІМЕНІ ІВАНА ПУЛЮЯ**

Циклова комісія програмних систем і комплексів

МЕТОДИЧНІ ВКАЗІВКИ

до виконання лабораторних робіт

з дисципліни

«Web-технології та web-дизайн»

напрямок підготовки 6.050101 «Комп'ютерні науки»

спеціальність 5.05010101 «Обслуговування програмних систем і комплексів»

відділення електронних апаратів

Тернопіль – 2016 рік

Методичні вказівки для лабораторних робіт з дисципліни «Web-технології та web-дизайн» для студентів спеціальності 5.05010101 «Обслуговування програмних систем і комплексів» / В.В. Довгань. – Тернопіль: ТК ТНТУ ім. І.Пулюя, 2016. – 103с.

Укладач: Довгань Володимир Вячеславович, викладач

Методичні вказівки розглянуто і схвалено на засіданні циклової комісії програмних систем і комплексів.

Протокол № 1 від « 30 » 08 2016 р.

Голова циклової комісії програмних систем і комплексів _____ Г.Я. Марціаш
« 30 » 08 2016 року

ЗМІСТ

1. Лабораторна робота №1	
HTML. Створення простої web-сторінки та її форматування, створення списків.....	4
2. Лабораторна робота №2	
HTML.Створення web-сторінки з таблицями, гіперпосиланнями та динамічними ефектами.....	11
3. Лабораторна робота №3	
HTML. Елементи форми Лабораторна робота №3.....	13
4. Лабораторна робота №4	
Створення статичного Web-сайту засобами мов XHTML та CSS.....	15
5. Лабораторна робота №5	
Розробка динамічних об'єктів засобами бібліотеки jQuery. Ajax.....	38
6. Лабораторна робота №6	
Створення лабораторного середовища шляхом інсталяції операційної системи Linux або на основі оболонки Denweg та модуля phpMyAdmin.....	43
7. Лабораторна робота №7	
Хостинг та FTP-доступ – основні поняття, можливості, інструменти.....	50
8. Лабораторна робота №8	
Основні конструкції мови сценаріїв (скриптів) PHP.....	60
9. Лабораторна робота №9	
Оператори умовного переходу та циклів у PHP.....	67
10. Лабораторна робота №10	
Створення та заповнення реляційної бази даних в технології PHP- MySQL.....	82
11. Лабораторна робота №11	
Методи автоматизації проектування та розробки WEB-сайтів.....	95

Лабораторна робота № 1

Тема: «HTML. Створення простої web-сторінки та її форматування, створення списків»

Мета: Уміти створювати прості web-сторінки та надавати їм певного вигляду, використовуючи для цього основні команди мови *HTML*.

Теоретичні відомості

Основні теги

Тег означення **початок та кінець** *html-файлу*

<html>*web-сторінка***</html>**

Пара тегів які описують **заголовок документа**

<head>*текст***</head>**

<title>*текст***</title>** - заголовок Windows-вікна.

Введення коментарів

<!-- *текст-коментар* **-->**

або у середині парного тега

<comment> *текст-коментар* **</comment>**

Введення тексту на сторінку

<body background="шлях/адреса файла зображення для тла"

bicolor = "колір тла" **text =** "колір символів" **>**

Текст

</body>

Теги форматування символів тексту

**** *текст* ****

товстий шрифт тексту

<i> *текст* **</i>**

шрифт- *курсив*

<u> *текст* **</u>**

підкреслений шрифт

_{ *текст* **}**

нижній індекс.

^{ *текст* **}**

верхній індекс

<big> *текст* **</big>**

великий шрифт

<small> *текст* **</small>**

малий шрифт

Створення абзаців

<p [align={left/right/center/justify}]>

Означає початок нового абзацу.

Наступне після тега **<p>** речення починатиметься з нового, вирівняного до лівого краю, абзацу без відступу. Між абзацами буде порожній рядок

**
** Наступний за цим тегом текст буде наведено у новому рядку без пропуску рядка

Заголовки

<h№>*заголовок***</h№>** - де № - це число яке змінюється від 1 до 6, визначає розмір символів Заголовка (за зменшенням). Починається з абзацу та вирівнюється до лівого краю.

Теги вирівнювання

<center>*текст по центру***</center>**

<left>*текст до лівого краю (за умовчанням)***</left>**

<right>*текст до правого краю***</right>**

Рисуння горизонтальної лінії

<hr width ="довжина у пікселях" **size=**"товщина лінії" **color=**"колір лінії"**>**

Шрифти

текст

розміри символів шрифту можуть бути від 1 до 7. (Розмір 3 вважається стандартним, він орієнтовно відповідає 10 пунктам).

Тег з фідформатованим текстом

<pre>текст відображатиметься у браузері так як записаний у коді (перехід на рядки, пропуски тощо)</pre>

Скорочення

<acronym title="текст розшифровки"> скорочення</acronym>

Основні кольори

<i>black</i> - чорний	#000000	<i>green</i> - зелений	#008000
<i>navy</i> - темно-синій	#000080	<i>teal</i> - бірюзовий	#008080
<i>silver</i> - срібний	#C0C0C0	<i>lime</i> - яскраво-зелений	#C0FF00
<i>blue</i> - синій	#0000FF	<i>aqua</i> - блакитний	#00FFFF
<i>maroon</i> - малиновий	#800000	<i>olive</i> - темно-зелений	#808000
<i>purple</i> - бузковий	#800080	<i>gray</i> - темно-сірий	#808080
<i>red</i> - червоний	#FF0000	<i>yellow</i> - жовтий	#FFFF00
<i>fuchsia</i> - рожевий	#FF00FF	<i>white</i> - білий	#FFFFFF

Службові та непечатні символи

<	<
>	>
 	Неразривний пробіл
&	&
« » "	« » “
–	-
—	--- подовжене тире
π	π

Створення списків

Маркований список:

<lh> заголовок списку</lh>
<ul [type={disc|circle|square}]>
 елемент списку
 елемент списку . . .

Нумерований список:

<lh>заголовок списку</lh>
<ol type="значення параметра" [start={0|1}]>
<li [type="значення параметра"]>елемент списку
елемент списку . . .

де значення параметра **type** задають зображення нумерації списку:

"i" - римськими малими (i, ii, iii, iv, ...),
"I" - римськими великими (I, II, III, IV, ...) цифрами,
"a" - латинськими малими (a, b, c, d, ...),
"A" - латинськими великими (A, B, C, ...) літерами.

Список визначень

<lh>Заголовок</lh>
<dl>
<dt> термін
 <dd> визначення 1
 <dd> визначення 2 ...
</dl>

Створення таблиць

<table параметри>

[**<tc>** заголовок таблиці **</tc>**]

<tr><th>текст заголовок у клітинці**</th> <td>**текст у клітинці**</td> ... </tr>**

<tr><th>текст заголовок у клітинці**</th> <td>**текст у клітинці**</td> ... </tr>**

.....

</table>

Порожню клітинку описують, як

<td></td> або **<td> </td> .**

Деякі параметри **<table>**

bordercolor = "колір рамки"

bgcolor = "колір тла"

border=товщина рамки

rowspan=n

n – кількість об'єднаних послідовних

colspan=n

клітинок у рядку (у стовпці) у відповідних

тегах **<th>** чи **<td>**

Графічні об'єкти

**

Гіперпосилання

Гіперпосилання на файл.

<a href = "адреса файлу"

name="#позначка"

taget={_blank|_self}>

<!-- вікно для тексту – у новому або у тому самому-- >

текст-гіперпосилання .

Гіперпосилання на деяке графічне зображення.

** **

Гіперпосилання в межах сторінки.

Визначається місце позначки у файлі

та гіперпосилання на створену позначку

** текст гіперпосилання **

Змінити колір гіперпосилання – у тегах **<BODY>**:

link = "колір" колір гіперпосилання

vlink="колір" змінює колір гіперпосилання після першого його використання

alink="колір" змінює колір активізованого гіперпосилання

Динамічні ефекти

<marquee height = висота смуги в пікселях

bgcolor = "колір тла смуги"

loop= число кількості проходів>

текст, що рухається

</marquee>

Навігаційна карта

Опис самого зображення карти має обов'язковий параметр:

****,

карту у цілому описується

<map name = "#назва карти">

<area параметри>

<!--для опису окремих зон зображення -->

...

</map>

параметри тега **<area>**:

href = "адреса ресурсу, який викликають"

alt = "альтернативний текст-підказка"

taget="{_blank|_self}" *<!-- вікно для тексту – у новому або у тому самому-- >*

shape="{rect/circ/poly/default}" *<!--тип форми області (прямокутником за умовчанням, колом, багатокутником, все зображення).*

coords = "список координат області"

Створення сайтів

Для поділу вікна броузера на фрейми

<frameset {cols|rows}="розмір лівої обл.,розмір правої обл.,...">

<frame параметри>

<frame параметри>

...

</frameset>

Розміри областей задають: відносно у % або фіксовано - у пікселях та розднюються ,.

Параметри **<frame >**

src=url стартового *html*-файлу

name=*назва фрейма* - значенням якого є назва фрейму, цю назву придумує користувач, вона використовуватиметься далі як значення параметра **target** тега **<a>**.

параметри для тегів **frameset** та **frame**

frameborder="{0|1}" *<!--задає наявність рамок фреймів (0 за умовчанням) або no/yes*

border =*ширину межі між фреймами у пікселях*

bordercolor=*колір межі (за умовчанням – сірий)*

тільки для frame

marginheight=*величину відступів зверху у пікселях*

marginwidth=*величину відступів від бокових*

scrolling="{no/yes/auto}" задає наявність чи відсутність смуг прокручування у вікні

noresize (забороняє змінювати розміри фрейму у вікні броузера (*без значень*))

Форми

Тег для опису форми

<form action="url обробочика даних"

method="{ get|post}" *<!--спосіб передачі даних по посиланню або за значенням*

enctype="{application/x-www-form-urlencoded або text/plain}" - тип кодування для обробки програмою або для передачі по електронній пошті

[target=значення]>

елементи форми

</form>

Поле введення: тексту, пароля, вибір файла, сритий текст:

<input type={text | password | file | hidden }

name=ім'я елемента

value=текст за умовчанням

size=кількість видимих символів рядка

maxlength=максимальна кількість символів рядка>

Кнопки: посилання даних, кнопки-картинки, відміни:

<input type={submit | image | reset}

value=назва на кнопку>

Список варіантів

<input type=checkbox name=значення1 [checked]>

<input type=checkbox name=значение2 >

.....

<input type=checkbox name=значениен>

список перемикачів:

<input type=radio name=значение1 [checked]>

<input type=radio name=значение2>

.....

<input type=radio name=значениен>

Список, що випадає:

<select name=ім'я списка

size=висота видимої частини списка (у рядках)

multiple (без значень) >

- вибір декількох пунктів

<option value=»передаєме значення»

selected (без значень)>

за умовчанням

.....

<option параметрин>

</select>

Текстове поле:

<textarea name=ім'я

rows=висота (у рядках)

cols=довжина (у символах)>

Текст

</textarea>.

Послідовність виконання роботи

1. Відкрийте редактор **NotePad**. Як це зробити?

2. Створіть за допомогою текстового редактора *html-файл* з особистими даними о собі.

Текст повинен мати заголовок, складатися з декількох абзаців, мати коментарі. Який *тег* використовується для введення тексту?

3. Задайте назву вікна *web-сторінки*. Який *тег* це робить?

<html>

<!--це файл file1.htm -->

<head>

<title>Особисті данні</title>

</head>

<body>

Я Сідор Сідорович Сідоренко. Народився 12 квітня 198...р.

у м.Одеса.

В 200... році закінчив школу №... м.Одеса.

Моя адреса: індекс, місто, вулиця,будинок,квартира.

</body>

</html>

4. Збережіть його під назвою *file11.htm* у власній папці.

5. Відкрийте файл *file1.htm* у браузері **Internet Explorer**.

Для цього відкрийте свою робочу папку і двічі клацніть мишею на назві файлу.

6. Відредагуйте сторінку. Для того щоб відредагувати файл, треба спочатку відкрити програму **NotePad**, а потім файл і кожного разу після редагування треба зберегати файл.

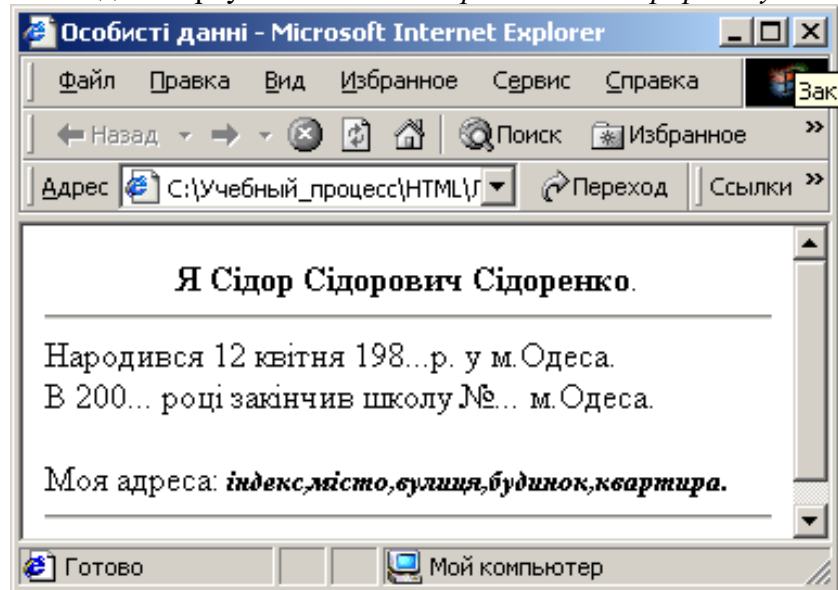
Задайте колір фону та тексту. Змінювайте відповідні параметри тега **body** - **bgcolor** і **text** (назви кольорів: *red, green, white, yellow, blue, #ffaa55* тощо).

7. Виконайте форматування тексту у файлі *file1.htm*. Застосуйте у тексті різні накреслення літер (жирний, курсив, підкреслений).

Заголовок тексту відцентруйте та відокремте від іншого тексту порожнім рядком.

Кожний абзац розташуйте з нового рядка.

У кінці всього тексту наведіть черту. *Які теги використали для форматування?*

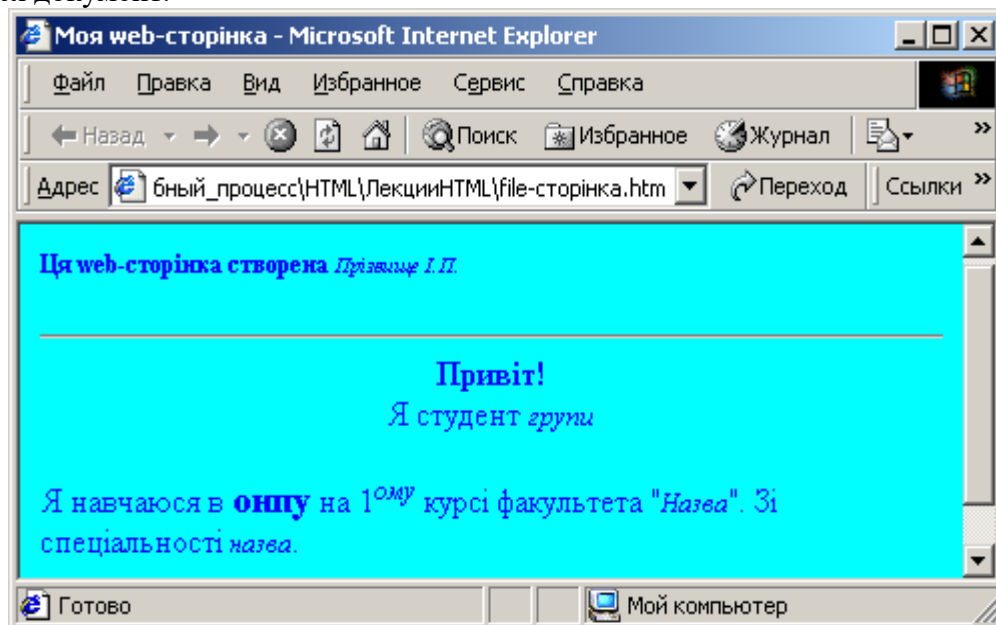


8. Створіть ще один *html-файл* з розповіддю про себе.

Поекспериментуйте з тегами форматування тексту. Використайте якнайбільше тегів форматування і надайте своїй сторінці якнайліпшого вигляду.

9. Збережіть файл із назвою *file2.htm*.

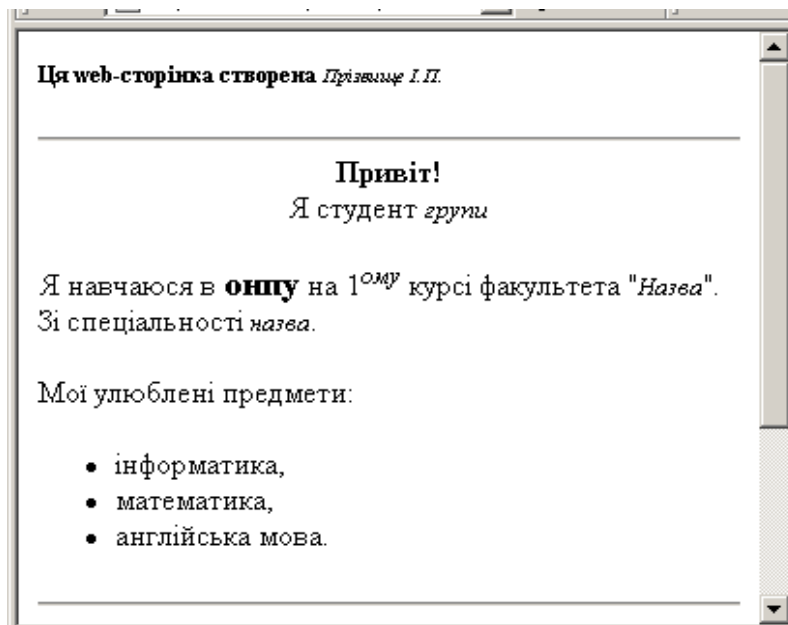
10. Перегляньте цей файл за допомогою броузера і поекспериментуйте з розмірами вікна, в якому демонструється документ.



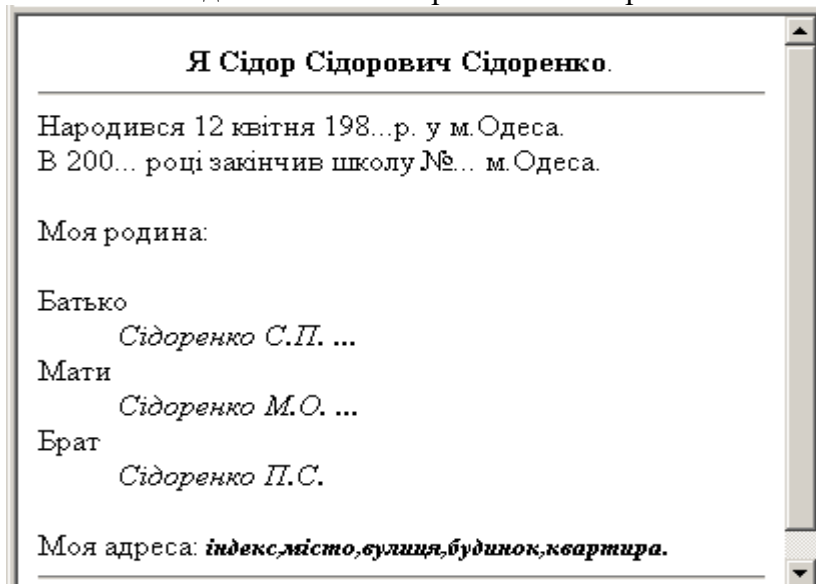
11. Удоскональте свою попередню *web-сторінку* та додайте до тексту *список своїх уподобань*.

12. Відкрийте *file2.htm* із розповіддю про себе. Список уподобань створіть як нумерований чи нелінійний список. *Чим відрізняється створення цих списків?*

13. Збережіть файл на диску і перегляньте його у броузері.



14. Проведіть на сторінці лінії різної ширини, довжини та кольорів. Як це зробити?
 15. Виокремте список у тексті іншим шрифтом. Який тег змінює тип шрифту?
 16. Додайте список тлумачення.
- Відкрийте *file1.htm* із особистими даними та створить список про своїх близьких.



Контрольні питання

1. Що таке web-документ?
2. Для чого призначена програма-броузер?
3. Що таке web-вузол (web-сайт)?
4. Яка структура простого web-документа?
5. Для чого призначена мова **html**?
6. Що таке тег і які є теги?
7. Які параметри може мати тег **body**?
8. Який тег позначає початок нового абзацу?
9. Які теги позначають товстий, курсивний і підкреслений шрифти?
10. Які теги призначені для вирівнювання елементів на сторінці?
11. Яке призначення тега **font**, параметри тега **font**?
12. Які є типи списків?
13. Як створити нумерований список?
14. Як створити нумерований список?
15. Як створити список означень?

Лабораторна робота № 2

Тема: «HTML.Створення web-сторінки з таблицями, гіперпосиланнями та динамічними ефектами»

Мета: Уміти додавати на сторінку таблицю, графічні та відіо-зображення, створювати динамічні ефекти, використовувати теги для створення різних гіперпосилань.

Послідовність виконання роботи

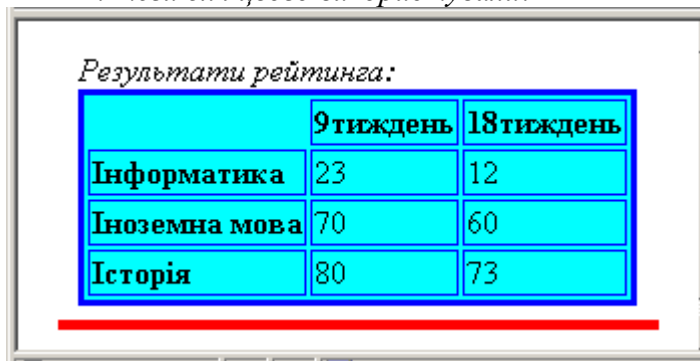
1. На новій web-сторінці створить таблицю з даними про свій рейтинг.

Таблиця буде складатися з 3 стовпців та 4 рядків.

Задайте заголовок таблиці “Результати рейтинга”.

Задайте заголовки стовпців та рядків.

Заповніть таблицю оцінками. Які теги для цього використовували?



	9тиждень	18тиждень
Інформатика	23	12
Іноземна мова	70	60
Історія	80	73

2. Новий файл збережить під назвою *file3.htm*. Перегляньте його у браузері.

3. Змініть вигляд таблиці.

Задайте товщину рамки таблиці, наприклад, 3, задайте кольори рамки та фона таблиці. Як це зробили?

4. Вирівнюйте текст у клітинках таблиці.

У заголовках – по центру, в інших клітинках – на ваш смак.

5. Поекспериментуйте з параметрами тега **table**. Об'єднайте деякі дві клітинки таблиці в одну.

Удоскональте свою сторінку якнайліпше.

6. Створіть ще одну таблицю.

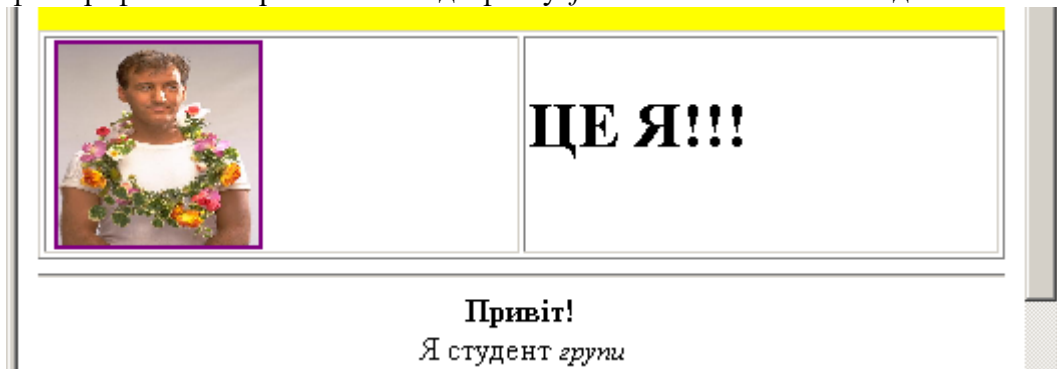
Відкрийте свою web-сторінку *file2.htm* та створіть таблицю для розміщення фотографії та тексту.

Таблиця повинна складатися з одного рядка та двох стовпців: у першому - фото, у другому – текст, наприклад, “Це я!”. (Графічні файли повинні мати розширення *bmp, jpg, gif*). З допомогою якого тега?

7. Відредагуйте графічне зображення..

Встановіть розміри фотографії, візьміть її у рамку. Як це зробили?

8. Зробіть фотографію як гіперпосиланням до файлу *file1.htm* з особистими даними. Як це зробити?



9. Змініть таблицю.

Зробіть її без рамки. З допомогою якого параметра це можна зробити ?

10. Вставте у свій файл гіперпосилання на html-файл, наприклад, на файл з таблицею.

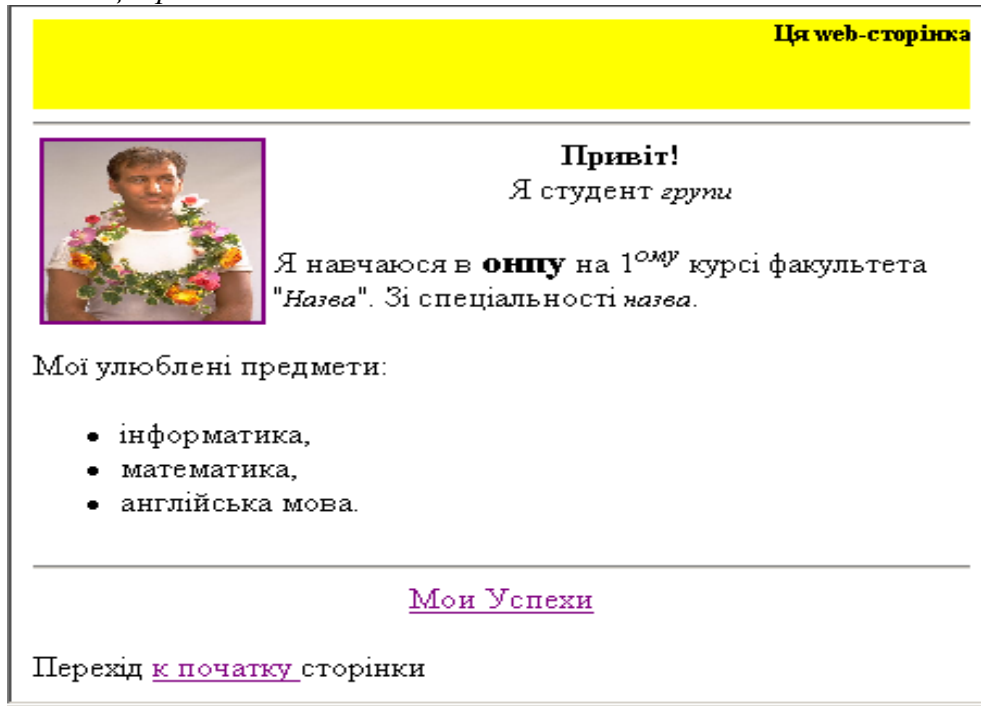
У кінці сторінки після черти введіть речення: *Мої успіхи* і зробіть його гіперпосиланням на файл з таблицею *file3.htm*. Як це зробили?

11. Створіть гіперпосилання у вигляді графічного зображення.

Клацнувши на *web*-сторінці на фотографії, повинен відкриватися файл з особистими даними *file1.htm*. Як це зробили?

12. Застосуйте гіперпосилання для переходу на початок сторінки.

В кінці сторінки розташуйте рядок з гіперпосиланням: *Перехід к початку сторінки* для переходу у початок сторінки. Як це зробити?



13. Застосуйте до заголовку динамічний ефект.

У файлі *file2.htm* зробіть рядок *Ця сторінка створена Прізвище І.П.* рухомим у смузі будь-якого коліру. Який тег для цього використовується?

Контрольні питання

1. Яке призначення тега **table**, параметри тега **table**?
2. Які теги формують у таблиці рядки, клітинки-заголовки і звичайні клітинки?
3. Як у таблиці об'єднати декілька клітинок в одну?
4. Який параметр використовують для вирівнювання елементів?
5. Яка особливість пари тегів `<pre>...</pre>`?
6. Як вставити графічне зображення у *web*-сторінку?
7. Які параметри може мати тег **img**?
8. За допомогою якого тега вставляють гіперпосилання?
9. Яке призначення параметрів **link**, **vlink**, **alink**?
10. Як деяке графічне зображення зробити гіперпосиланням?
11. Які є види посилань у межах одної *web*-сторінки?
12. Що таке якори?
13. Як запустити звук одночасно з відкриванням сторінки?
14. Як створити динамічний ефект рухомого тексту?
15. Які ви знаєте параметри рухомого тексту?

Лабораторна робота № 3

Тема: “HTML. Елементи форми”.

Мета: Вміти створювати сторінки з кнопками, списками, полями введення тощо, оформляти сторінки, використовуючи таблиці стилів.

Послідовність виконання роботи

1. Скористайтесь існуючим сайтом . Додайте до нього ще одну сторінку *Анкета*.
2. Створіть сторінку *Анкета.htm*. Ведений текст на сторінці не форматуйте, тільки зробіть заголовок №1 та курсив де потрібно.
3. Для введення *Імені* додайте *Поле введення* з текстом-підказкою у рядку *ІІІІ*.
4. Додайте *Поле* для введення особистого *паролю* , який буде складатися з 5 символів.
5. Розташуйте *Список перемикачів* з 2 елементів для опису *статі*.
6. Для ознаки діяльності створіть *Список варіантів* з 5 елементів, розташованих по різному. З обраним за умовчанням елементом *Студент..*
7. Створіть *Відкриваючий список* з 5 елементів (наприклад, *науково-пізнавальна, фантастика, фентезі, пригоди, історичні*), 3 елемента з яких є видимими. Останній елемент (*історичні*) оберіть за умовчанням. Передбачити багатоваріантний вибір.
8. Розташуйте *Текстове поле* з 5 рядків та 40 символів вздовж, з впливаючою підказкою *Смелее!* .
9. У кінці сторінки додайте дві кнопки. Першу *Кнопку відміни* з підписом *Отмена*, другу *Кнопку Подачи* з підписом *Готово*.
10. Додайте гіперпосилання на якусь адресу.

Анкета

Заполни анкету для вступления в клуб "Виртуальная библиотека"

Введите имя:
ФИО

Введите пароль:

Пол: ☐ Мужской ☐ Женский

Род занятий:

☒ Студент
☐ Школьник
☐ Аспирант
☐ Работаю ☐ Не работаю

Какие книги любишь читать:

Что читаешь сейчас?

Поделись впечатлениями о прочитанном:

Начинай здесь

Дополнительная информация на сайте: <http://www.Bibl.kiev.ua>

Контрольні питання

1. Для чого використовуються форми?

2. З допомогою яких тегів створюються форми?
3. Чим відрізняються ведення різних варіантів Кнопки?
4. Як ввести пороль?
5. У чому різниця між описом елементів списку перемикачів та списку варіантів?

Лабораторна робота №4

Тема: Створення статичного Web-сайту засобами мов XHTML та CSS

Метою лабораторної роботи є оволодіння навичками створення статичних Web-сайтів у відповідності до загальноприйнятих стандартів.

Загальне завдання на лабораторну роботу:

1. Вивчити особливості розмітки Web-документів із використанням мови XHTML та CSS.
2. Розробити макет та виконати кодування сторінок статичного Web-сайту відповідно до варіанта.
3. Виконати інформаційне (графічне та текстове) наповнення сторінок Web-сайту.

Деталізоване завдання полягає у наступному:

1. Згідно з макетом головної сторінки Web-сайту виконати її розмітку, а саме, кодування мовами XHTML та CSS. Для розмітки забороняється використовувати таблиці, лише базові теги: `<div>`, ``.
2. Меню сайту створювати за допомогою списків (``, ``) з відповідним форматуванням CSS.
3. Розмітку документу виконувати згідно із правилами створення XHTML-документів (із синтаксисом XML). Перевірити відповідність стандартам XML на сайті <http://validator.w3.org>.
4. Текстові та графічні фрагменти сторінок заповнювати обов'язково.
5. Заголовок сторінки оформлювати у вигляді зображення або текстової назви.
6. Нижній колонтитул має містити інформацію про студента та дату створення сайту.
7. З кожної сторінки має бути доступний перехід на головну сторінку.
8. Максимальна кількість кольорів та шрифтів на сайті – 4.
9. CSS-стилі створити у окремому файлі, який підключається до усіх сторінок сайту.

Основні теоретичні відомості

Мова розмітки Web-документів XHTML

HTML – мова розмітки гіпертекстових документів (Web-документів, Web-сторінок) призначена для вирішення декількох задач: структурування інформації на Web-сторінці, логічного об'єднання Web-сторінок у єдиний сайт завдяки гіперпосиланням, а також форматування окремих елементів сторінки (тексту, зображень тощо). Слід зауважити, що вирішення останньої задачі на сьогоднішній день вирішується завдяки використанню спеціалізованої мови стильового оформлення Web-документів CSS. Хоча HTML містить засоби так званого «фізичного» форматування фрагментів документа (декорування тексту, вирівнювання елементів тощо), використовувати їх не рекомендується, оскільки це значно ускладнює процес модифікації дизайну сторінки, а також призводить до недоцільного дублювання HTML-коду. Особливу увагу слід звернути на те, що HTML – не є мовою програмування, ця мова не містить управляючих конструкцій типу циклів або умовних операторів. В той же час HTML містить спеціальні інструкції для включення в сторінку програм мовою Javascript, або інших об'єктів, які можуть бути виконані для даної сторінки.

Мова XHTML – це модифікація HTML для підтримки сумісності з правилами формування XML-документів. Ця модифікація дозволяє використовувати семантику мови HTML (призначення інструкцій залишається таким самим) та строгий та однозначний синтаксис XML, який гарантує відсутність синтаксичних помилок у вихідному коді сторінки, а також можливість відображення того ж інформаційного наповнення сторінки на різних пристроях, зокрема, персональних, кишенькових комп'ютерах або мобільних телефонах завдяки засобам автоматичної трансформації XML-документів.

Розглянемо синтаксис XHTML. Документ XHTML є текстовим файлом, який за своєю структурою складається з елементів, що визначають правила форматування його фрагментів. Кожний

елемент записується як початкова частина, вміст та кінцева частина (рис.4.1). Початкова та кінцева частини визначаються тегами. Тег – це запис виду `<назва>` для початкової частини та `</назва>` - для кінцевої, де *назва* відповідає призначенню елемента.

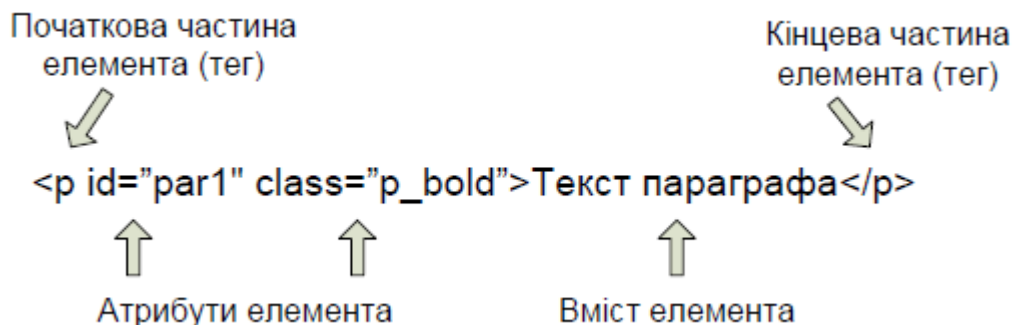


Рисунок 1.1. Структура елемента XHTML

З рис.4.1 видно, що початкова частина елемента може містити додаткові параметри – атрибути, які уточнюють деякі характеристики елемента в залежності від призначення елемента, наприклад, для запису `Гіперпосилання` атрибут `href` вказує на цільову адресу гіперпосилання.

Якщо вміст елемента відсутній, допустимим є запис виду: `<назва/>`. Наприклад, `
`, `<p/>`, `<hr/>`.

Елементи сторінки об'єднуються у строгий ієрархічний порядок, причому кожний XHTML-документ повинен мати обов'язкову базову структуру (рис.1.2):

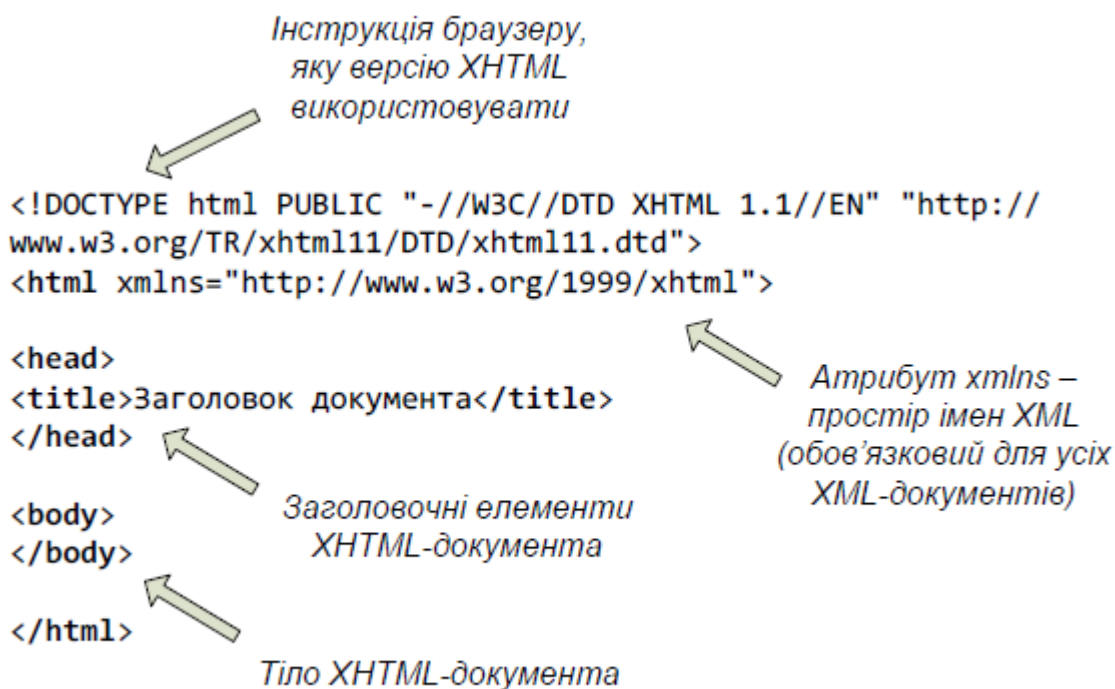


Рисунок 4.2. Мінімальний вміст XHTML-документа

Важливою особливістю базової структури XHTML-документа є наявність в ньому інструкції `<!DOCTYPE>`, яка вказує браузеру, який саме діалект мови використав автор сторінки і, відповідно, що саме вважати помилками на сторінці. На практиці використовують декілька варіацій HTML та XHTML. Переважно їх відмінності полягають у забороні використання фреймів та застарілих тегів «фізичного» форматування тексту (наприклад, ``, `<center>`). Проте усі діалекти XHTML мають задовольняти правилам побудови так званих коректно сформованих (well-formed) XML-документів.

Найголовніші правила полягають у наступному:

1. Документ обов'язково повинен мати один або більше елементів.
2. В документі має бути один і лише один кореневий елемент. Решта елементів ієрархічно підпорядковуються йому.
3. Елементи мають бути ієрархічно коректно підпорядковані між собою.
4. Елементи мають бути записаними в нижньому регістрі.
5. Елементи мають включати початкову та кінцеву частину (рис.4.1).
6. Назви атрибутів елементів мають бути в нижньому регістрі.
7. Значення атрибутів мають бути записані у подвійних лапках.

В обов'язковій секції <head> документа, як правило, записують елементи посилань на зовнішні файли, зокрема, програм мовою Javascript: `<script type="text/javascript" src="myfile.js"></script>` або файлів стильового оформлення CSS: `<link rel="stylesheet" type="text/css" href="theme.css" />`

Крім того, в даній секції можуть бути мета-теги, що описують сам документ, наприклад, заголовок, автор, ключові слова для пошукової системи тощо.

В обов'язковій секції <body> міститься тіло документа, власне те, що бачить користувач: текст документа, посилання на зовнішні графічні та інші види мультимедійних ресурсів, а також посилання на програмні модулі.

Розглянемо основні типи елементів HTML (табл.4.1).

Таблиця 4.1 Групи елементів XHTML

Група елементів	Опис	Приклади елементів
1	2	3
Базові	Блокові контейнери: тіла документа, абзаців, заголовків, а також коментарі.	<code><body></code> , <code><p></code> , <code><h1>..<h6></code> , <code>
</code> , <code><hr/></code> , <code><div></code> <code><!--коментар --></code>
«Фізичне» форматування	Рядкові елементи декорування тексту: шрифту, кольору, інтервалів тощо.	<code><s></code> , <code></code> , <code><i></code> , <code><sub></code> , <code><sup></code> , <code><u></code> , <code><q></code> , <code><strike></code> , <code></code>
Елементи форм	Описують елементи введення даних від користувача (кнопки, поля введення, перемикачі тощо)	<code><form></code> , <code><input></code> , <code><textarea></code> , <code><select></code> , <code><option></code> , <code><label></code> , <code><button></code> , <code><fieldset></code>
Фрейми	Елементи відображення декількох сторінок у одному вікні браузера	<code><frame></code> , <code><frameset></code> , <code><noframe></code> , <code><iframe></code>
Зображення	Вказують на зовнішні файли зображень (jpg, gif тощо)	<code></code> , <code><map></code> , <code><area></code>

Посилання зовнішні файли	на	Дозволяють додавати гіперпосилання, включати зовнішні програми мовою Javascript, файли стилів CSS та інші	<a>, <link>
Списки		Організують текст у вигляді списків (нумерованих або декорованих)	, ,
Таблиці		Структурують дані в табличну форму	<table>, <tr>, <td>, <th>
1		2	3
Стили CSS		Застосовують стилі для документа	<style>
Програмування		Застосування можливостей управління сторінкою, зокрема, підключення Flash	<script>, <noscript>, <object>, <param/>
Елементи секції <head>		Описують службову інформацію про сторінку: автор, ключові слова, назва, опис та інші	<meta>, <title>, <base>

Одиниці виміру, прийняті в XHTML

Для виміру розмірів, відступів, інтервалів тощо можуть бути застосовані абсолютні та відносні одиниці виміру (табл.4.2).

Таблиця 4.2

Одиниці виміру XHTML

Відносні	Абсолютні
<p><i>px</i> – піксель (залежить від пристрою виведення)</p> <p>% – відсоток від можливого розміру <i>em</i> – розмір відносно ширини стандартної літери «m» <i>ex</i> – розмір відносно висоти стандартної літери «x»</p>	<p><i>in</i> – дюйми <i>cm</i> – сантиметри</p> <p><i>mm</i> – міліметри <i>pt</i> – пункти (1/72 дюйма) <i>pc</i> – піка (12 пунктів)</p>

Атрибути елементів XHTML

Атрибути визначають додаткові характеристики елементів в залежності від призначення того чи іншого елемента. Проте існує група атрибутів, які є допустимими майже до будь-яких з них (табл.4.3). Виключення складають елементи: <base>, <head>, <html>, <meta>, <param>, <script>, <style> та <title>.

Таблиця 4.3 Базові атрибути XHTML

Назва атрибута	Призначення	Приклади
id	Унікальний ідентифікатор елемента. Застосовується при визначенні стилю CSS або при здійсненні доступу через Javascript	<code><p id= p1 >text</p></code>
style	Дозволяє визначити стиль CSS для поточного елемента	<code><h1 style= color:black >чорний заголовок </h1></code>
title	Описує додаткову інформацію про елемент	<code><div title= Меню ></code> ... <code></div></code>
class	Задає клас CSS, до якого належить елемент	<code><div class= bold-text ></code> Текст <code></div></code>

Колір у XHTML задається декількома способами:

1. Англomовні ідентифікатори для основних тонів (black, white, red, green, yellow тощо). Наприклад, `<body bgcolor="white"> ... </body>`
2. 16-кове значення компонент у кольоровому просторі RGB (червоний-зелений-синій), наприклад: `#ff0000` (червоний), `#0000ff` (синій), `#cccccc` (сірий).

Адресація ресурсів у XHTML

Для унікальної ідентифікації ресурсів прийнято використовувати URL-схему. Структура URL має наступний вигляд:

`<схема>://<логін>:<пароль>@<хост>:<порт>/<URL-шлях>?<параметр1>=<значення1>&<параметр2>`

`=<значення2>...#<id елем.>`, де

`<схема>` – назва протоколу, зокрема http, https, ftp тощо (обов'язковий компонент);

`<логін>` – ім'я користувача;

`<пароль>` – пароль користувача;

`<хост>` – доменне ім'я сервера або його IP-адреса (обов'язковий компонент); `<порт>` – TCP-порт;

`<URL-шлях>` – локальна адреса ресурсу на сервері відносно кореневого каталогу Web-сервера, заданого в конфігурації;

`<параметр1>=<значення1>` – додаткові параметри запита (може бути декілька);

`<id елем.>` – елемент сторінки, який буде активним на сторінці (вказується значення атрибута *id* елемента).

Деякі приклади URL-адресації:

`http://www.example.com/index2.html#p123`
`http://www.example.com/path/to/file/page1.php?a=1&b=abc&c=q1`
`http://10.111.123.231:8080/page/to/resource/`

Крім того, допустимим є використання так званих відносних шляхів URL:

`./index3.html`, де `«./»` означає, що ресурс знаходиться у поточному

каталозі;

`/file.php`, де `«/»` означає, що `file.php` необхідно адресувати з кореневого каталогу сайту;

`../img/i1.jpg`, де `«../»` означає, що для знаходження файлу `i1.jpg` треба піднятися на один каталог вище, і потім шукати файл у каталозі `img`.

Відносну адресацію рекомендується використовувати для знаходження локальних ресурсів. Це спрощує подальшу реструктуризацію сайту.

Мова CSS

Призначенням CSS є застосування властивостей форматування елементів у єдиний стандартизований спосіб та відокремлення засобів структурування елементів (XHTML) від їх візуального подання. Стили CSS можна задати декількома способами:

1. Як атрибут `style` елемента. В такому випадку властивості форматування будуть застосовані лише для даного елемента. Використання цього способу на практиці можна вважати виключенням, оскільки він майже тотожний фізичному форматуванню засобами HTML. Наприклад: `<p style="color:red">Червоний текст</p>`.

2. Як елемент `<style>` записаний, як правило, у секції `<head>`. Такий спосіб задає стилі, що будуть застосовані для всього даного документа. Цей спосіб характерний для форматування деякої окремої сторінки, яка стилізована інакше, ніж решта. Наприклад:

```
<head> <style> p {color:red;
margin:5px} #div1 {border: 1px red
solid}
</style>

</head>
```

3. Використання директиви `@import` мови CSS. Цей спосіб дозволяє розділяти інструкції CSS на окремі файли (модулі), зокрема, відокремлюючи окремі секції сайту: заголовки, основний вміст, меню тощо. Наприклад:

```
<style>

@import url("menu.css")

@import url("content.css")

</style>
```

4. Як зовнішній файл CSS, що підключається завдяки елементу `<link>` для усіх сторінок сайту. Цей спосіб є найбільш вживаним, оскільки дозволяє формувати усі сторінки єдиним способом, скорочує обсяги HTML-сторінок та відокремлює структуру документа від засобів її відображення. Наприклад:

```
<link rel="stylesheet" type="text/css" href="main.css" />
```

Пріоритетність застосування стилів у випадку наявності декількох способів віддається першому способу, інакше другому, інакше третьому, інакше четвертому. Якщо жодний зі стилів не визначено для елемента, підставляються налаштування за замовчуванням, прийняті в браузері.

Синтаксис CSS

Синтаксично інструкції CSS, які називають селекторами, записуються у вигляді правил, що застосовуються для одного, групи або усіх елементів сторінки. Приклад:



Рисунок 4.3. Приклад запису правила CSS

Селектори

Основою CSS є селектори, які дозволяють визначати один елемент, групу елементів або усі елементи сторінки, для яких буде застосовано те чи інше правило форматування. Існують десятки правил, які дозволяють визначати цільові елементи форматування, причому допустимими є правила як CSS, так і правила, що описані в стандарті мови XPath (спеціально розробленої для пошуку елементів у XML-документах). Наведемо приклади найбільш вживаних селекторів.

1. «*» застосувати для усіх елементів сторінки:

* {margin: 10px}

2. «#X» застосувати для елемента з атрибутом *id* рівним X:

#myid {border: 0}

У XHTML: <p id="myid">Text</p>

3. «.X» застосувати для елементів, в яких атрибут *class* має значення X:

.myclass {top: 15px} У

XHTML:

<div class="myclass">Text1</div>

<p class="myclass">Text2</p>

4. «X Y» застосувати для Y, який знаходиться ієрархічно всередині X (як безпосередньо під X, так і глибше за ієрархією):

ul a {text-decoration: none}

У XHTML:

 <a>Link

5. «X» застосувати для усіх елементів, назва тегу яких є X.

p {font-weight: bold}

6. «X, Y, Z» застосувати для всіх перерахованих селекторів: p, a, #myid, .myclass {border: 1px solid black}

7. «X:visited», «X:hover» та інші. Застосувати для селектора X зі специфічним станом: посилання, яке вже відвідане, курсор миші над елементом тощо. Запис після двокрапки називається псевдокласом. a: visited {color: #ffffff}

8. «X+Y» застосувати для Y, який знаходиться на тому ж рівні ієрархії та розміщено безпосередньо за X.

h1+p {padding: 5px} У

XHTML: <h1>Z1</h1>

`<p>P1</p>`

9. «X>Y» застосувати для Y, який знаходиться ієрархічно *безпосередньо* всередині X (порівняти з п. 4):

`ul a {text-decoration: none}`

У XHTML:

` <a>Link`

Наступні правила запозичені з мови XPath.

10. «X[Y]» застосувати для X, у якого наявний атрибут Y:

`h4[title] {color: 0}`

У XHTML:

`<h4 title="abcd">Header</h4>`

11. «X[Y=|V|]» застосувати для X, у якого атрибут Y має значення V:

`a[href="example.com"] { text-decoration: none }`

12. «X[Y\$=|V|]» застосувати для X, у якого значення атрибута Y закінчується на V:

`a[href$=".jpg"] { text-decoration: none }`

Етапи розробки статичного Web-сайту

На практиці процес створення статичного Web-сайту спрощено можна розділити на такі етапи.

1. Розробка структури Web-сайту.

На даному етапі проводиться аналіз задач, для вирішення яких призначено Web-сайт, формується список сторінок, які складатимуть інформаційне наповнення сайту та відповідну систему навігації. В рамках лабораторної роботи список сторінок та структурні (ієрархічні) зв'язки між ними задано попередньо, але в реальній ситуації необхідно узгодити структуру сайту із замовником в залежності від інформаційного наповнення, яке планується розмістити на Web-сайті.

2. Створення графічного дизайну сайту, що включає розробку макетів головної та підпорядкованих сторінок у вигляді прямокутних блоків, які відповідають функціональному призначенню секції сторінки (наприклад, меню, заголовок, інформаційний блок тощо). Крім цього, визначаються стилеві особливості сайту: кольорова схема, набір шрифтів, їх розміри для різних структурних елементів сторінки та інші елементи дизайну з урахуванням питань зручності експлуатації (usability). В лабораторній роботі ці параметри задаються варіантом.

3. Кодування сторінок сайту мовою XHTML та CSS. На цьому етапі власне і відбувається створення сторінок у вигляді XHTML та CSS файлів, які визначають візуальне подання сторінки. При виконанні лабораторної роботи необхідно дотримуватись загальноприйнятих стандартів: задовольняти правилам формування структурних елементів сторінок, слідувати за коректним іменуванням елементів (атрибут *id*), використанням DOCTYPE та мета-тегів заголовку сторінки. Крім того, сторінки мають проходити валідацію W3C (<http://validator.w3.org>).

4. Наповнення сторінок сайту інформаційним вмістом. На даному етапі формується текстовий та мультимедійний вміст сторінки (графіка, відео тощо).

5. Тестування. На цьому етапі виконується тестування системи навігації, гіперпосилань, розмітки сторінки, інформаційного наповнення, а також зручності експлуатації.

Кодування сторінок сайту мовою XHTML та CSS

Згідно з наведеними вище етапами фактичним завданням на лабораторну роботу є виконання третього пункту, а також частково другого (для розробки структури підлеглих сторінок сайту). Тому розглянемо більш детально процес підготовки та власне кодування сторінок сайту, зупинившись на створенні головної сторінки. Вхідною інформацією є макет сторінки, наприклад, як на рис.4.4:

Головне меню	
Заголовок	
Основний вміст	Бічна панель
Нижній колонтитул	

Рисунок 4.4. Макет сторінки

Будемо вважати, що сайт має фіксовані розміри у вікні браузера і його буде розміщено у центрі по горизонталі. Встановимо розміри кожного із структурних елементів сторінки.

1. Головне меню: ширина – 800 пікселів (далі рх), висота – 50рх.
2. Заголовок: ширина – 800рх, висота – 150рх.
3. Основний вміст: ширина – 500рх, висота змінюється в залежності від наповнення.
4. Бічна панель: ширина – 300рх, висота змінюється в залежності від наповнення.
5. Нижній колонтитул: ширина – 800рх, висота – 70рх.

Наступним кроком є створення шаблону XHTML-сторінки, який містить визначення документу DOCTYPE, параметри кодування, а також інші метатеги, необхідні для застосування пошуковими системами (рис.1.5). Слід зауважити, що у заголовку шаблону присутній елемент `<link>`, який підключає зовнішній файл стилів *main.css*, з папки *css*. Цей файл містить налаштування форматування елементів сторінки, його необхідно використовувати для усіх сторінок сайту для підтримки єдиного стилю відображення (шрифти, кольори тощо).

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-type" content="text/html; charset=UTF-8" />
<title>Назва сторінки</title>
<meta http-equiv="Content-Language" content="uk" />

<meta name="description" content="Опис" />
<meta name="keywords" content="Ключові слова" />

<meta name="author" content="Студент" />

<link type="text/css" media="all" rel="stylesheet" href="css/main.css" />
</head> <body>
</body>
</html>

```

Рисунок 4.5. Шаблон XHTML-сторінки

Оскільки сторінки сайту мають фіксовану ширину (800px), доцільним є створення контейнера для решти елементів сторінки. Тому в середину тегу `<body>` внесемо наступний запис:

```

<body>
<div id="container">

Вміст контейнера

</div>
</body>

```

Контейнер реалізовано як елемент *div* (універсальний блок) з унікальним ідентифікатором *container*. У файл *main.css* внесемо запис виду:

```

#container {
    width: 800px;
    margin: auto; border: 1px solid
black
}

```

Цим записом встановлено ширину контейнера 800 пікселів, границю блока, а також горизонтальне вирівнювання «по центру». Саме властивість *margin* (поле) зі значенням *auto* вказує, що поля треба встановлювати автоматично з усіх боків – за замовчуванням це визначає центральне вирівнювання. При цьому висота буде обраховуватись в залежності від вмісту блока (рис.4.6).



Рисунок 4.6. Контейнер сторінки

З рис.4.6 видно, що верхній бік контейнера має відступ від вікна браузера, щоб його позбавитись необхідно встановити відступи для елементів *html* та *body* в 0:

```

html, body {
    margin: 0; padding: 0;
}

```


Наступним кроком є ініціалізація основних блоків сторінки: головного меню, заголовку, основного вмісту, бічної панелі та нижнього колонтитула. Як правило, ці блоки записують тегами <div> у XHTML-сторінці всередині контейнера:

```
<body>
<div id="container">
    <div id="nav">Головне меню</div>
    <div id="header">Заголовок</div>
    <div id="sidebar">Бічна панель</div>
    <div id="content">Вміст</div>
    <div id="footer">Нижній колонтитул</div>
</div>
</body>
```

Додамо базові стилі для цих блоків у файл main.css:

```
#container {
    width: 800px;
    margin: auto;
}
#nav {
    height: 50px;
}
#header {
    height: 150px;
}
#sidebar {    float: right;
    width: 300px;
}
#footer {
    height: 70px;
}
div {    border: 1px solid
black;
}
```

Висоту меню та нижнього колонтитула встановлено в абсолютних одиницях. Для зручності сприйняття усі елементи <div> виводяться з рамкою (останній запис). Особливим блоком є *sidebar*, оскільки за завданням його треба розмістити праворуч від основного вмісту. Крім того, оскільки *sidebar* – це додаткова панель, текст основного вмісту має «обтікати» його збоку. Це досягається шляхом використання властивості *float: right*. Результат ілюструє рис.4.7.

Головне меню	
Заголовок	
Вміст	Бічна панель
Нижній колонтитул	

Рисунок 4.7. Результат форматування базових блоків сторінки У випадку, коли бічна панель буде за висотою більшою, ніж основний вміст, спостерігатиметься ефект накладання тексту бічної панелі на нижній колонтитул. Щоб позбавитись цього, необхідно відмінити «обтікання» для нижнього колонтитула в файлі CSS:

```
#footer {
    height: 70px; clear:
both;
}
```

Наступним кроком є створення головного меню. Для прикладу оберемо популярні в Web пункти: «Про автора», «Послуги», «Контакти». Для меню будемо використовувати список HTML:

```
<div id="nav">
    <ul>
        <li><a href="#">Про автора</a></li>
        <li><a href="#">Послуги</a></li>
        <li><a href="#">Контакти</a></li>
    </ul>
</div>
```

Застосуємо наступні правила CSS для меню:

```
#nav ul {
margin: 0 auto; padding: 2px 0 0 0; border-top:
1px dotted #AFAFAF; border-bottom: 1px
dotted #AFAFAF; text-align: center; width:
100%;
}
```

У наведеному блоці CSS встановлюються правила для списку (*ul*) всередині контейнера *#nav*: центральне вирівнювання, відступ зліва на 2px, верхню та нижню границю шириною 1px, сірим кольором пунктирним стилем. Ширина списку – 100%.

```
Визначимо стиль елементів списку li: ul li
{
```

```
list-style-type:none;
margin:0; padding:3px 0;
display:inline;
}
```

В цьому блоці CSS: маркер – відсутній, відступ – 0, поле зліва – 3px, тип елемента – рядковий (щоб розмістити список у вигляді рядка). #nav ul li a {

```
padding:0 20px; line-height:40px;
color:#afafaf; text-transform:uppercase;
font-weight:bold; text-decoration: none;
}
```

Для гіперпосилань меню встановлено відступ з правого та лівого боку 20px, колір – сірий, усі символи – у верхньому регістрі, напівжирні.

При наведенні та натисканні лівої кнопки миші колір тексту пункту меню буде замінено на темно-сірий:

```
#nav ul li a:hover, #nav ul li a.active
{
color: #333;
}
```

Вміст заголовку сторінки складається з наступного коду: <div id="header">Привітання
 Головна сторінка</div>

Стилеве оформлення CSS при цьому матиме вигляд:

```
#header { text-align: center;
font-size: 3em; color:
#AFAFAF; line-height: 1.1em;
letter-spacing: -0.04em;
background: white;
}
```

Особливістю наведеного блоку CSS є збільшений втричі шрифт, на 4% зменшена відстань між символами, та збільшена на 10% висота рядка. Ілюструє вигляд головного меню та заголовку рис. 4.8:

ПРО АВТОРА ПОСЛУГИ КОНТАКТИ

Привітання Головна сторінка

Рисунок 4.8. Зовнішній вигляд меню та заголовку сторінки Наступний елемент сторінки – бічна панель, стилеве оформлення якої наведено вище. Вміст XHTML виглядатиме наступним чином:

```
<div id="sidebar">
```

```
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc lacinia, odio in viverra
varius, nisl quam laoreet leo, eu pretium nibh sapien eu diam. Etiam interdum lorem eget turpis
vehicula euismod. Phasellus vestibulum placerat risus fermentum mollis. Fusce eget pretium eros.
Aenean sed laoreet magna. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere
cubilia Curae;
```

```
</p>
```

```
</div>
```

Після бічної панелі розміщено основний вміст. Для прикладу візьмемо 2 інформаційні повідомлення. XHTML-код блоку основного вмісту наведено нижче:

```
<div id="content">
  <ul >
    <li>
      <a href="#" class="info">
        <h2>Інформаційне повідомлення №1</h2>
        <p>Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem
        Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer
        took a galley of type and scrambled it to make a type specimen book.
        </p>
      </a>
    </li>
    <li>
      <a href="#" class="info">
        <h2>Інформаційне повідомлення №2</h2>
        <p>Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem
        Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer
        took a galley of type and scrambled it to make a type specimen book.
        </p>
      </a>
    </li>
  </ul>
</div>
```

Контейнер *content* містить список, елементи якого складаються з гіперпосилання, заголовку та тексту повідомлення.

Відповідні стилі вмісту наведено нижче: a.info { background: url('../images/info.gif') no-repeat left center; }

Даним правилом встановлюємо фон гіперпосилання.

```
h2 { font-size: 1.4em;
font-weight: bold; color:
#333; margin: 10px;
}
```

Зовнішній вигляд заголовку *h2* включає налаштування шрифту, кольору та відступу, які розглядалися вище.

Наступним правилом знімаємо підкреслення з гіперпосилання, перетворюємо його на блок, дещо збільшуємо розмір шрифту та робимо колір світлішим (в чорно-білій гамі).

#content LI a

```
{ text-decoration: none; color:#333;
padding:10px 0 10px 40px; display:block;
color:#afafaf; font-size:103%; border-
bottom: 1px dotted #CCC }
```

Зовнішній вигляд бічної панелі та основного вмісту наведено на рис.4.9.



Рисунок 4.9. Зовнішній вигляд блоків основного вмісту та бічної панелі

Останній елемент сторінки – нижній колонтитул. Як правило, його фон виділяють іншим кольором, а з інформаційної точки зору він містить інформацію про автора сторінки та, за необхідності, дублює деякі частини головного меню:

```
<div id="footer">
  <div id="altnav">
    <a href="#">Про автора</a> -
    <a href="#">Послуги</a> -
    <a href="#">Контакти</a>
  </div>
</div>
```

Стильове оформлення нижнього колонтитула:


```
#footer a
{
    color:#B86443;    margin-top:25px;
    display: inline-block;    font-size:90%;
}
```

При цьому гіперпосилання перетворено на рядковий елемент із властивостями блоку (*inline-block*), а шрифт зменшено на 10%.

Таким чином, сторінка у цілому виглядатиме як на рис.4.10:


Привітання Головна сторінка

Інформаційне повідомлення №1

 Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc lacinia, odio in viverra varius, nisl quam laoreet leo, eu pretium nibh sapien eu diam. Etiam interdum lorem eget turpis vehicula euismod. Phasellus vestibulum placerat risus fermentum mollis. Fusce eget pretium eros. Aenean sed laoreet magna. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae;

Інформаційне повідомлення №2

 Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book.

[Про автора](#) - [Послуги](#) - [Контакти](#)

Рисунок 4.10. Результат розмітки сторінки

Варіанти завдання

Роботу виконувати самостійно. Номер варіанта завдання визначити, взявши останні дві цифри номера залікової книжки студента (табл.4.3).

Таблиця 4.3 Варіанти завдань

№	Макет головної сторінки (Заголовок – місце логотипу або назви, Меню – навігація по сайту, Вміст – вміст сторінки, Додаткова секція – рекламний блок, Нижній колонтитул – контактна інформація)	Призначення сайту та набір сторінок (склад меню)	Фіксована чи плаваюча ширина сторінки								
1	2	3	4								
1.	<table><tr><td>Заголовок</td><td>Меню</td></tr><tr><td colspan="2">Вміст</td></tr><tr><td colspan="2">Додаткова секція</td></tr><tr><td colspan="2">Нижній колонтитул</td></tr></table>	Заголовок	Меню	Вміст		Додаткова секція		Нижній колонтитул		Портфоліо автора: 1. Про автора 2. Контакти 3. Основні роботи 4. Корисні посилання	фіксована
Заголовок	Меню										
Вміст											
Додаткова секція											
Нижній колонтитул											
2.	<table><tr><td>Заголовок</td><td>Меню</td></tr><tr><td>Вміст</td><td>Додаткова секція</td></tr><tr><td colspan="2">Нижній колонтитул</td></tr></table>	Заголовок	Меню	Вміст	Додаткова секція	Нижній колонтитул		Сайт-«візитка» компанії: 1. Про компанію 2. Види продукції 3. Контакти 4. Історія створення	фіксована		
Заголовок	Меню										
Вміст	Додаткова секція										
Нижній колонтитул											
3.	<table><tr><td>Заголовок</td><td>Меню</td></tr><tr><td>Додаткова секція</td><td>Вміст</td></tr><tr><td colspan="2">Нижній колонтитул</td></tr></table>	Заголовок	Меню	Додаткова секція	Вміст	Нижній колонтитул		Персональна сторінка студента: 1. Біографія 2. Інформація про дисципліни в університеті 3. Розклад занять 4. Досягнення	фіксована		
Заголовок	Меню										
Додаткова секція	Вміст										
Нижній колонтитул											
4.	<table><tr><td>Заголовок</td><td>Меню</td></tr><tr><td rowspan="2">Вміст</td><td>Додаткова секція</td></tr><tr><td>Нижній колонтитул</td></tr></table>	Заголовок	Меню	Вміст	Додаткова секція	Нижній колонтитул	Персональна сторінка викладача: 1. Біографія 2. Інформація про дисципліни 3. Розклад занять викладача 4. Список наукових публікацій	плаваюча			
Заголовок	Меню										
Вміст	Додаткова секція										
	Нижній колонтитул										

1	2	3	4									
5.	<table><tr><td colspan="2">Заголовок</td></tr><tr><td colspan="2">Меню</td></tr><tr><td>Вміст</td><td>Додаткова секція</td></tr><tr><td colspan="2">Нижній колонтитул</td></tr></table>	Заголовок		Меню		Вміст	Додаткова секція	Нижній колонтитул		Сайт про програмний продукт: 1. Опис програмної розробки 2. Сторінка завантаження із зображеннями - прикладами роботи 3. Контакти автора 4. Сторінка зворотного зв'язку	фіксована	
Заголовок												
Меню												
Вміст	Додаткова секція											
Нижній колонтитул												
6.	<table><tr><td colspan="2">Заголовок</td></tr><tr><td colspan="2">Меню</td></tr><tr><td rowspan="2">Вміст</td><td>Додаткова секція</td></tr><tr><td>Нижній колонтитул</td></tr></table>	Заголовок		Меню		Вміст	Додаткова секція	Нижній колонтитул	Інформаційний сайт про культурну подію: 1. Інформація про концерт 2. Новини культурного життя 3. Сторінка із розташуванням місця проведення 4. Інформація про виконавця	плаваюча		
Заголовок												
Меню												
Вміст	Додаткова секція											
	Нижній колонтитул											
7.	<table><tr><td colspan="3">Заголовок</td></tr><tr><td rowspan="2">Меню</td><td rowspan="2">Вміст</td><td>Додаткова секція</td></tr><tr><td>Нижній колонтитул</td></tr></table>	Заголовок			Меню	Вміст	Додаткова секція	Нижній колонтитул	Сайт про котеджне містечко: 1. Про містечко 2. Фотоальбом будівництва 3. Розташування на карті 4. Соціальні програми	плаваюча		
Заголовок												
Меню	Вміст	Додаткова секція										
		Нижній колонтитул										
8.	<table><tr><td colspan="2">Заголовок</td><td rowspan="4">Меню</td></tr><tr><td colspan="2">Вміст</td></tr><tr><td colspan="2">Додаткова секція</td></tr><tr><td colspan="2">Нижній колонтитул</td></tr></table>	Заголовок		Меню	Вміст		Додаткова секція		Нижній колонтитул		Персональна сторінка викладача: 1. Біографія 2. Інформація про дисципліни 3. Розклад занять викладача 4. Список наукових публікацій	фіксована
Заголовок		Меню										
Вміст												
Додаткова секція												
Нижній колонтитул												
9.	<table><tr><td>Меню</td><td>Заголовок</td></tr><tr><td colspan="2">Вміст</td></tr><tr><td>Додаткова секція</td><td>Нижній колонтитул</td></tr></table>	Меню	Заголовок	Вміст		Додаткова секція	Нижній колонтитул	Сайт-«візитка» компанії: 1. Про компанію 2. Види продукції 3. Контакти 4. Історія створення	плаваюча			
Меню	Заголовок											
Вміст												
Додаткова секція	Нижній колонтитул											

1	2	3	4								
10.	<table><tr><td rowspan="4">Меню</td><td>Заголовок</td></tr><tr><td>Вміст</td></tr><tr><td>Додаткова секція</td></tr><tr><td>Нижній колонтитул</td></tr></table>	Меню	Заголовок	Вміст	Додаткова секція	Нижній колонтитул	Персональна сторінка студента: 1. Біографія 2. Інформація про дисципліни в університеті 3. Розклад занять 4. Досягнення	фіксована			
Меню	Заголовок										
	Вміст										
	Додаткова секція										
	Нижній колонтитул										
11.	<table><tr><td>Заголовок</td><td>Меню</td></tr><tr><td colspan="2">Вміст</td></tr><tr><td colspan="2">Додаткова секція</td></tr><tr><td colspan="2">Нижній колонтитул</td></tr></table>	Заголовок	Меню	Вміст		Додаткова секція		Нижній колонтитул		Портфолію автора: 1. Про автора 2. Контакти 3. Основні роботи 4. Корисні посилання	плаваюча
Заголовок	Меню										
Вміст											
Додаткова секція											
Нижній колонтитул											
12.	<table><tr><td>Заголовок</td><td>Меню</td></tr><tr><td>Додаткова секція</td><td>Вміст</td></tr><tr><td colspan="2">Нижній колонтитул</td></tr></table>	Заголовок	Меню	Додаткова секція	Вміст	Нижній колонтитул		Сайт-«візитка» компанії: 1. Про компанію 2. Види продукції 3. Контакти 4. Історія створення	плаваюча		
Заголовок	Меню										
Додаткова секція	Вміст										
Нижній колонтитул											
13.	<table><tr><td>Меню</td><td>Заголовок</td></tr><tr><td colspan="2">Вміст</td></tr><tr><td>Додаткова секція</td><td>Нижній колонтитул</td></tr></table>	Меню	Заголовок	Вміст		Додаткова секція	Нижній колонтитул	Портфолію автора: 1. Про автора 2. Контакти 3. Основні роботи 4. Корисні посилання	фіксована		
Меню	Заголовок										
Вміст											
Додаткова секція	Нижній колонтитул										
14.	<table><tr><td colspan="2">Заголовок</td></tr><tr><td colspan="2">Меню</td></tr><tr><td>Вміст</td><td>Додаткова секція</td></tr><tr><td colspan="2">Нижній колонтитул</td></tr></table>	Заголовок		Меню		Вміст	Додаткова секція	Нижній колонтитул		Персональна сторінка студента: 1. Біографія 2. Інформація про дисципліни в університеті 3. Розклад занять 4. Досягнення	плаваюча
Заголовок											
Меню											
Вміст	Додаткова секція										
Нижній колонтитул											

Продовження табл. 1.3

1	2	3	4								
15.	<table><tr><td colspan="3">Заголовок</td></tr><tr><td rowspan="2">Меню</td><td rowspan="2">Вміст</td><td>Додаткова секція</td></tr><tr><td>Нижній колонтитул</td></tr></table>	Заголовок			Меню	Вміст	Додаткова секція	Нижній колонтитул	Сайт про котедрне містечко: 1. Про містечко 2. Фотоальбом будівництва 3. Розташування на карті 4. Соціальні програми	плаваюча	
Заголовок											
Меню	Вміст	Додаткова секція									
		Нижній колонтитул									
16.	<table><tr><td rowspan="4">Меню</td><td>Заголовок</td></tr><tr><td>Вміст</td></tr><tr><td>Додаткова секція</td></tr><tr><td>Нижній колонтитул</td></tr></table>	Меню	Заголовок	Вміст	Додаткова секція	Нижній колонтитул	Інформаційний сайт про культурну подію: 1. Інформація про концерт 2. Новини культурного життя 3. Сторінка із розташуванням місця проведення 4. Інформація про виконавця	фіксована			
Меню	Заголовок										
	Вміст										
	Додаткова секція										
	Нижній колонтитул										
17.	<table><tr><td>Заголовок</td><td>Меню</td></tr><tr><td colspan="2">Вміст</td></tr><tr><td colspan="2">Додаткова секція</td></tr><tr><td colspan="2">Нижній колонтитул</td></tr></table>	Заголовок	Меню	Вміст		Додаткова секція		Нижній колонтитул		Персональна сторінка студента: 1. Біографія 2. Інформація про дисципліни в університеті 3. Розклад занять 4. Досягнення	фіксована
Заголовок	Меню										
Вміст											
Додаткова секція											
Нижній колонтитул											
18.	<table><tr><td>Заголовок</td><td rowspan="4">Меню</td></tr><tr><td>Вміст</td></tr><tr><td>Додаткова секція</td></tr><tr><td>Нижній колонтитул</td></tr></table>	Заголовок	Меню	Вміст	Додаткова секція	Нижній колонтитул	Сайт про котедрне містечко: 1. Про містечко 2. Фотоальбом будівництва 3. Розташування на карті 4. Соціальні програми	плаваюча			
Заголовок	Меню										
Вміст											
Додаткова секція											
Нижній колонтитул											
19.	<table><tr><td colspan="2">Заголовок</td></tr><tr><td colspan="2">Меню</td></tr><tr><td>Вміст</td><td>Додаткова секція</td></tr><tr><td colspan="2">Нижній колонтитул</td></tr></table>	Заголовок		Меню		Вміст	Додаткова секція	Нижній колонтитул		Портфоліо автора: 1. Про автора 2. Контакти 3. Основні роботи 4. Корисні посилання	плаваюча
Заголовок											
Меню											
Вміст	Додаткова секція										
Нижній колонтитул											

1	2		3	4						
20.	<table><tr><td>Меню</td><td>Заголовок</td></tr><tr><td colspan="2">Вміст</td></tr><tr><td>Додаткова секція</td><td>Нижній колонтитул</td></tr></table>		Меню	Заголовок	Вміст		Додаткова секція	Нижній колонтитул	Інформаційний сайт про культурну подію: 1. Інформація про концерт 2. Новини культурного життя 3. Сторінка із розташуванням місця проведення 4. Інформація про виконавця	фіксована
Меню	Заголовок									
Вміст										
Додаткова секція	Нижній колонтитул									
21.	<table><tr><td rowspan="4">Меню</td><td>Заголовок</td></tr><tr><td>Вміст</td></tr><tr><td>Додаткова секція</td></tr><tr><td>Нижній колонтитул</td></tr></table>		Меню	Заголовок	Вміст	Додаткова секція	Нижній колонтитул	Інформаційний сайт про культурну подію: 1. Інформація про концерт 2. Новини культурного життя 3. Сторінка із розташуванням місця проведення 4. Інформація про виконавця	фіксована	
Меню	Заголовок									
	Вміст									
	Додаткова секція									
	Нижній колонтитул									
22.	<table><tr><td>Заголовок</td><td>Меню</td></tr><tr><td>Додаткова секція</td><td>Вміст</td></tr><tr><td colspan="2">Нижній колонтитул</td></tr></table>		Заголовок	Меню	Додаткова секція	Вміст	Нижній колонтитул		Персональна сторінка викладача: 1. Біографія 2. Інформація про дисципліни 3. Розклад занять викладача 4. Список наукових публікацій	плаваюча
Заголовок	Меню									
Додаткова секція	Вміст									
Нижній колонтитул										
23.	<table><tr><td>Заголовок</td><td rowspan="4">Меню</td></tr><tr><td>Вміст</td></tr><tr><td>Додаткова секція</td></tr><tr><td>Нижній колонтитул</td></tr></table>		Заголовок	Меню	Вміст	Додаткова секція	Нижній колонтитул	на сторінка студента: 1. Біографія 2. Інформація про дисципліни в університеті 3. Розклад занять 4. Досягнення	плаваюча	
Заголовок	Меню									
Вміст										
Додаткова секція										
Нижній колонтитул										

1	2			3	4	
24.	Меню	Заголовок		Сайт про програмний продукт: 1. Опис програмної розробки 2. Сторінка завантаження із зображеннями - прикладами роботи 3. Контакти автора 4. Сторінка зворотного зв'язку	фіксована	
		Вміст				
		Додаткова секція				
		Нижній колонтитул				
25.	Меню	Заголовок		Персональна сторінка студента: 1. Біографія 2. Інформація про дисципліни в університеті 3. Розклад занять 4. Досягнення	фіксована	
		Вміст				
		Додаткова секція				
		Нижній колонтитул				
26.	Меню	Заголовок		Персональна сторінка студента: 1. Біографія 2. Інформація про дисципліни в університеті 3. Розклад занять 4. Досягнення	плаваюча	
		Вміст				
		Додаткова секція				
		Нижній колонтитул				
27.	Меню	Вміст	Заголовок		Портфоліо автора: 1. Про автора 2. Контакти 3. Основні роботи 4. Корисні посилання	Фіксована
			Додаткова секція			
			Нижній колонтитул			
28.	Меню	Заголовок		Сайт-«візитка» компанії: 1. Про компанію 2. Види продукції 3. Контакти 4. Історія створення	плаваюча	
		Вміст				
		Додаткова секція				
		Нижній колонтитул				

1	2		3	4								
29.	<table><tr><td>Меню</td><td>Заголовок</td></tr><tr><td colspan="2">Вміст</td></tr><tr><td>Додаткова секція</td><td>Нижній колонтитул</td></tr></table>		Меню	Заголовок	Вміст		Додаткова секція	Нижній колонтитул	Персональна сторінка викладача: 1. Біографія 2. Інформація про дисципліни 3. Розклад занять викладача 4. Список наукових публікацій	фіксована		
Меню	Заголовок											
Вміст												
Додаткова секція	Нижній колонтитул											
30.	<table><tr><td>Заголовок</td><td>Меню</td></tr><tr><td colspan="2">Вміст</td></tr><tr><td colspan="2">Додаткова секція</td></tr><tr><td colspan="2">Нижній колонтитул</td></tr></table>		Заголовок	Меню	Вміст		Додаткова секція		Нижній колонтитул		Сайт про програмний продукт: 1. Опис програмної розробки 2. Сторінка завантаження із зображеннями - прикладами роботи 3. Контакти автора 4. Сторінка зворотного зв'язку	плаваюча
Заголовок	Меню											
Вміст												
Додаткова секція												
Нижній колонтитул												

Контрольні запитання

1. Визначення елемента XHTML та його атрибуту.
2. Правила формування коректно сформованого XML-документа.
3. Визначення DOM-моделі документа.
4. Недоліки використання таблиць XHTML для створення розмітки сторінки.
5. Універсальні атрибути XHTML.
6. Поняття блокової моделі CSS.
7. Основні селектори CSS.

Лабораторна робота №5

Тема: Розробка динамічних об'єктів засобами бібліотеки jQuery. Ajax.

Метою лабораторної роботи є вивчення основ роботи з бібліотеки jQuery при реалізації практичних задач Web-програмування.

Завдання на лабораторну роботу:

1. Ознайомитись із синтаксисом та базовими можливостями бібліотеки jQuery.
2. Набути практичних навичок маніпулювання вмістом Web-сторінок.
3. Вивчити способи реалізації інтерактивного інтерфейсу користувача за допомогою засобів JQuery Ajax.

Основні теоретичні відомості

Бібліотека JQuery як інструмент розробки інтерактивних Web- додатків

Активний розвиток застосування різноманітних Web-додатків, зокрема, соціальних мереж та засобів електронної комерції в Internet, зумовив необхідність створення інтерактивних інтерфейсів користувача, які за функціональними можливостями наближуються до традиційних віконних додатків. Місце програмного інструментарію для вирішення цієї задачі за останні 5-7 років зайняли різноманітні бібліотеки на основі вбудованої в Web- браузер мови Javascript. Серед таких засобів найбільш поширеною та вдалою виявилась бібліотека JQuery. Вона поєднує в собі лаконічний синтаксис, надзвичайно розвинуті засоби пошуку елементів Web-сторінки на основі CSS та XML-орієнтованої спеціалізованої мови XPath, функції маніпуляції XHTML DOM, анімаційні ефекти, а також ефективний інструментарій асинхронного обміну інформацією з Web-сервером Ajax.

Базовий синтаксис команди JQuery

Оскільки бібліотека JQuery написана мовою Javascript, то застосовувати код можна всюди, де допустимий код Javascript: в оброблювачах подій, викликаючи функції тощо. Спрощено запис команди JQuery виглядає наступним чином:

`$(selector).action1(args1).action2(args2) ...`

або

`$.action(args)`

де знак \$ означає скорочену назву об'єкта JQuery. Цей знак можна замінити на виклик: JQuery(selector);

selector визначає один або групу елементів Web-документа, об'єднаних за певним критерієм (назвою тега, ідентифікатору, CSS-класу тощо);

action, action1, action2 – визначають функцію, що треба застосувати для відібраних селектором елементів (встановити стиль CSS, заповнити/стерти вміст елемента, додати/вилучити оброблювач деякої події тощо);

arg, arg1, arg2 – необов'язкові аргументи функцій, що застосовуються до елементів (новий вміст елемента, його атрибуту тощо).

Якщо вказано декілька дій (*action*) до відібраних селекторів, то вони виконуються послідовно, наприклад:

`$("p").html("abcd").css("color", "green")`

В цьому прикладі для усіх параграфів буде внесено текст `-abcd` та встановлено колір тексту в зелений.

Селектори jquery

Бібліотека JQuery надає широкий спектр можливостей щодо пошуку елементів Web-сторінки з використанням синтаксису CSS та XPath. Наприклад, щоб застосувати деяке правило CSS для усіх елементів, достатньо виконати команду: `$('*').css('color','blue')`. Інші приклади:

`$("input:checked").length` – повертає кількість елементів форм, які виділені;

`$("td:empty").text("порожній").css('background', 'red')` – знаходить усі комірки таблиці `<td>` за умови, що вміст їх порожній (псевдоклас *empty*), після цього вносить в ці комірки текст «порожній» та встановлює для них колір фону в червоний.

`$("#d1,#s1,#par24").css('border','1px dashed black')` – знаходить елементи з ідентифікаторами *d1*, *s1*, *par24* та встановлює для них пунктирну границю розміром 1px чорним кольором;

`$("input[value='321']").val("123")` – повертає усі елементи форм, в які уведено значення «321» і замінює це значення на «123»;

`$("div[id]").text("My text")` – знаходить усі елементи `<div>`, що містять атрибут *id* та заповнює їх вміст текстом «My text».

Методи уточнення та обробки результатів відбору елементів JQuery

Після відбору елементів засобами селекторів, бібліотека JQuery має можливість додатково відфільтрувати результат:

а) *children()* – отримати усіх нащадків для знайдених елементів:

`$("#d1").children().css('background-color', 'red')` знаходить усіх нащадків елемента з ідентифікатором *d1* та застосовує для них правило CSS;

б) *filter(x)* – відфільтрувати елементи, використовуючи замість «x» функцію, інший селектор або елемент JQuery:

```
$('p').filter(function(index) { return
    $(this).text().length > 3;
}).css('background-color', 'red')
```

В наведеному прикладі знаходяться усі параграфи, до них додатково застосовується функція фільтрації, яка перевіряє довжину тексту всередині. Якщо довжина більша трьох, то для таких параграфів буде застосовано правило CSS;

в) *each()* – виконати ітерацію по усіх знайдених елементах:

```
$('div').each(function(index) { alert(index + ': '
+ $(this).text());
});
```

В даному прикладі буде знайдено усі елементи `<div>` та для кожного з них застосовується функція, яка виведе номер елемента та його вміст.

г) *map()* – застосувати функцію для кожного зі знайдених елементів:

```
$(':input').map(function() { return this.id; }).get().join(',');
```

В цьому прикладі буде знайдено усі елементи форм, в кожного з них буде взято ідентифікатор і з них буде сформовано рядок виду: *id1,id2,id3*; де *id1*,*id2*,*id3* – ідентифікатори елементів.

Маніпуляція вмістом Web-документа

До групи методів маніпуляції елементами сторінки відносяться: додавання/вилучення властивостей CSS, встановлення/очищення текстового вмісту елемента та його дочірніх елементів тощо.

1) *css(property)*, *css(property,value)* – отримує та встановлює значення властивості CSS відповідно:

```
var a = $ („#e1“).css („position“) // взяти значення position для елемента з
```

id=e1

`$(„p“).css(„position“,a)` // встановити усім параграфам значення властивості `position`, як у змінній `a`

2) `html()`, `html(value)` – отримує та встановлює значення HTML-вмісту, включаючи теги та вміст дочірніх елементів, відповідно:

`alert$(„#a1“).html())` // вивести вміст елемента з `id=a1`

`$(„#a1“).html(„123“)` // встановити вміст елемента з `id=a1` значенням `123` (створити елемент `` всередині `a1`)

3) `remove()` - вилучити знайдені елементи:

`$(„c1, #e1“).remove();` // вилучити наступні елементи: у яких клас = `c1`; з `id=e1`

3) `empty()` - очистити знайдені елементи:

`$(„p“).empty();` // очистити вміст усіх параграфів

4) `after()`, `before()` – вставити вміст після/до знайдених елементів:

`$(„#d1“).after(„000“)` // вставити текст `000` після елемента з `id=d1`

`$(„#d1“).before(„Link“)` // вставити гіперпосилання перед елементом з `id=d1`

5) `text()`, `text(value)` – отримує та встановлює значення текстового вмісту, без тегів, але із текстовим вмістом дочірніх елементів, відповідно:

`alert$(„#a1“).text())` // вивести вміст елемента з `id=a1`

`$(„#a1“).text(„123“)` // встановити вміст елемента з `id=a1` значенням `123` (елемент `` не створюється, вставляється лише текст «`123`»)

Оброблювачі подій

Завдяки оброблювачам подій програміст має змогу застосовувати той чи інший програмний код, реагуючи на дії користувача (події від миші, клавіатури, операцій уведення/виведення). Розглянемо способи встановлення оброблювачів подій:

1) `bind(eventType [, eventData], handler(eventObject))` – універсальний метод встановлення оброблювача події до знайдених елементів, де `eventType` – тип події, яка оброблюється, `eventData` – необов'язковий параметр, який містить дані, що передадуться оброблювачу, `handler(eventObject)` – функція-оброблювач події:

`$(„p“).bind(„click“,function() { alert$(this.text()); });` // для усіх `<p>` встановити як оброблювач події натискання на ліву кнопку миші функцію, яка виводить вміст елемента

`$(„p“).bind(„click“,function(e){alert(e.pageX+';'+e.pageY)})` - // для усіх `<p>` встановити як оброблювач події натискання на ліву кнопку миші функцію, яка виводить координати курсору в точці, де відбулось натискання

Зауваження. „`e`“ – параметр, який передається будь-якому оброблювачу і уточнює подію, містить властивості про натиснуту клавішу, координати курсора, про тип елемента та інші.

2) `click()`, `dblclick()`, `focus()`, `blur()`, `change()`, `load()`, `unload()` – методи для обробки найпоширеніших подій: натискання та подвійного натискання кнопки миші, встановлення/втрати фокусу елемента, зміни його вмісту, завантаження/вивантаження елемента (зображення, сторінки).

`$(„input“).change(function() {
alert(„вміст змінено“);
});`

3) `unbind([eventType] [, handler(eventObject)])` – відміння використання оброблювача події,

де *eventType* – тип події, яка оброблюється, *handler(eventObject)* – функція-оброблювач події:
\$(‘div’).unbind(‘click’); // усім <div> відмінити обробку події *onClick*

```
var handler = function() {  
    alert('оброблювач');  
};  
  
$(‘#a1’).bind(‘click’, handler);  
  
$(‘#a1’).unbind(‘click’, handler); // handler – вказівник на функцію, що використовувались при  
bind
```

4) *\$(document).ready(handler)* – оброблювач події «завантажено DOM- модель».

Даний тип подій застосовується замість події *-onload* елемента <body>:

```
<body onload=“alert(1)”>... </body>
```

На відміну від *-onload* оброблювач *-ready* виконується раніше – в момент, коли сформовано дерево DOM об’єктів сторінки. В той же час *-onload* викликається пізніше, а саме після завантаження усіх сторонніх об’єктів сторінки: зображень, флеш-компонентів тощо. Приклад оброблювача

```
$(document).ready(function() { alert(„Сторінка  
    завантажена”)  
});
```

Асинхронний обмін інформацією з Web-сервером. Ajax

Ajax (асинхронний Javascript) – це підхід до створення інтерактивних Web-додатків, згідно з яким обмін даними між Web-браузером та Web-сервером відбувається у фоновому режимі та надає можливості оновлювати лише частину Web-сторінки. Такий підхід зменшує мережний трафік та дозволяє будувати додатки, наближені до «віконних», які є більш звичними для користувача. Головним чином підхід базується на використанні засобів мови Javascript. При цьому з серверного боку жодних змін виконувати не потрібно у порівнянні з традиційним підходом. На клієнті за Ajax відповідає об’єкт Javascript XMLHttpRequest. Передача даних за допомогою цього об’єкта може відбуватись як у синхронному, так і у асинхронному режимі. Складність безпосереднього використання цього об’єкта зумовлена неповною сумісністю реалізації в різних браузерах. Тому, як правило, використовують крос- платформні бібліотеки, які уніфікують доступ до даного об’єкта. Однією з таких бібліотек є JQuery. З огляду на задачі Ajax JQuery пропонує наступні можливості:

1. Завантаження даних з сервера безпосередньо у елемент сторінки (наприклад, у параграф) – метод *\$.load()*.
2. Завантаження та виконання програмного коду мовою Javascript – метод *\$.getScript()*.
3. Завантаження даних із сервера у форматі JSON, що спрощує перетворення даних у об’єкти Javascript – метод *\$.getJSON()*
4. Відправлення даних на сервер - метод *\$.post()*
5. Обробка помилок, що виникли при передачі даних.
6. Сериалізація даних HTML-форми для подальшої передачі на сервер.

```
$(<selector>).serialize()
```

Завдання

1. Реалізувати форму реєстрації користувача HTML
Форма повинна складатись з полів:
а. Ім’я

- b. Прізвище
 - c. Email
 - d. Пароль
 - e. Повторити пароль
 - f. Адреса
 - g. Кнопка реєстрації
2. За допомогою CSS розмістити поля в дві колонки.
 3. За допомогою JavaScript та jQuery реалізувати валідацію форми, перевірку полів:
 - a. Ім'я, прізвище, Email, пароль та повторити пароль – на заповненість поля
 - b. Email – на вірно вказаний Email
 - c. Пароль та повторити пароль – на їх рівність один одному, та на кількість символів більше 6.

Про помилки під час валідації чи її успішність вивести вверху форми відповідно червоним або зеленим кольором.

Лабораторна робота №6

Тема: створення лабораторного середовища шляхом інсталяції операційної системи Linux або на основі оболонки Denwer та модуля phpMyAdmin.

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 темами установки, управління та оновлення 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.в.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 віддаленій 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, транзакцій і кирилических кодувань (windows-1251);
- phpMyAdmin - панель керування базою даних MySQL, а також скрипт, 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:\WebServers.

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:\WebServers\home\localhost\www\, а робоча папка для баз даних MySQL – Z:\usr\local\mysql5\data\ або D:\WebServers\usr\local\mysql5\data\. Перегляд поточних 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 6.1-6.6 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 6.1 – Вікно вибору мови для Ubuntu

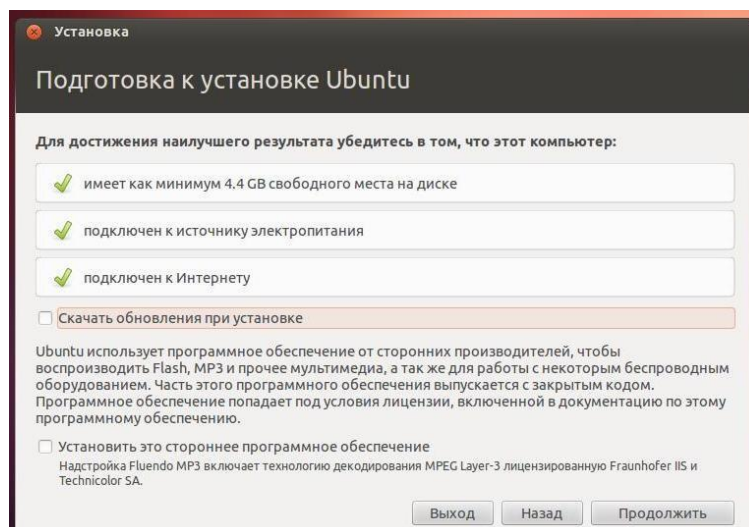


Рисунок 6.2 – Вікно підготовки до встановлення Ubuntu

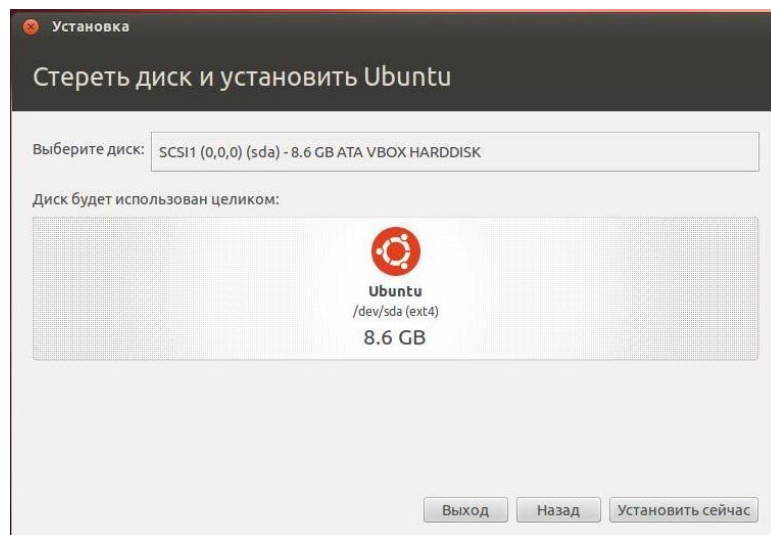


Рисунок 6.3 – Вікно встановлення Ubuntu на окремий жорсткий диск

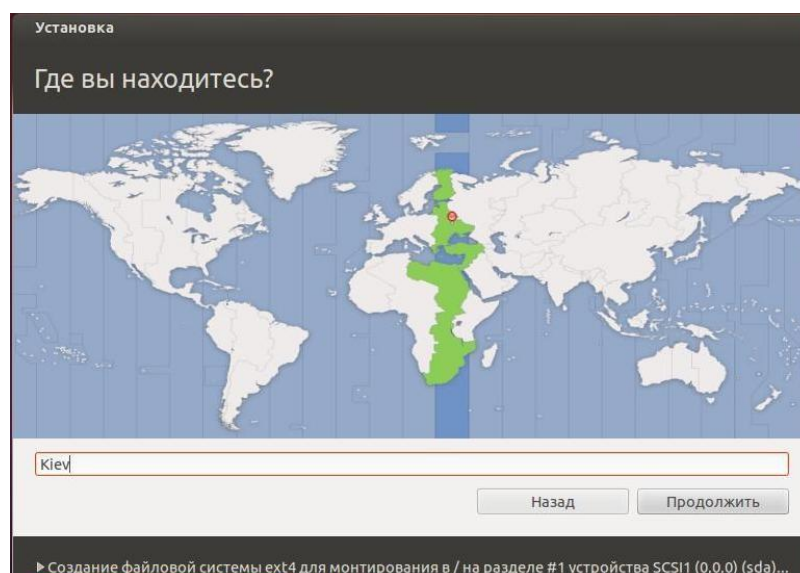


Рисунок 6.4 – Вікно вибору часового поясу для Ubuntu

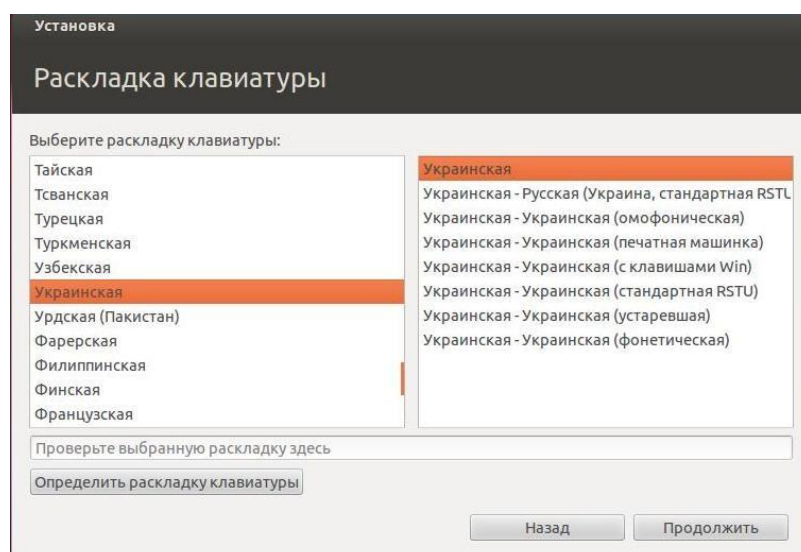


Рисунок 6.5 – Вікно вибору розкладки клавіатури для Ubuntu

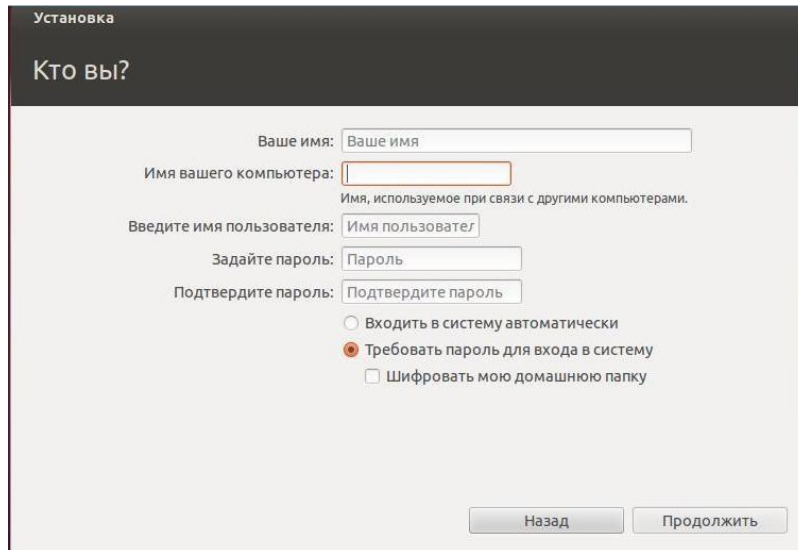



Рисунок 6.6 – Вікно введення персональних даних для Ubuntu

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 6.7).

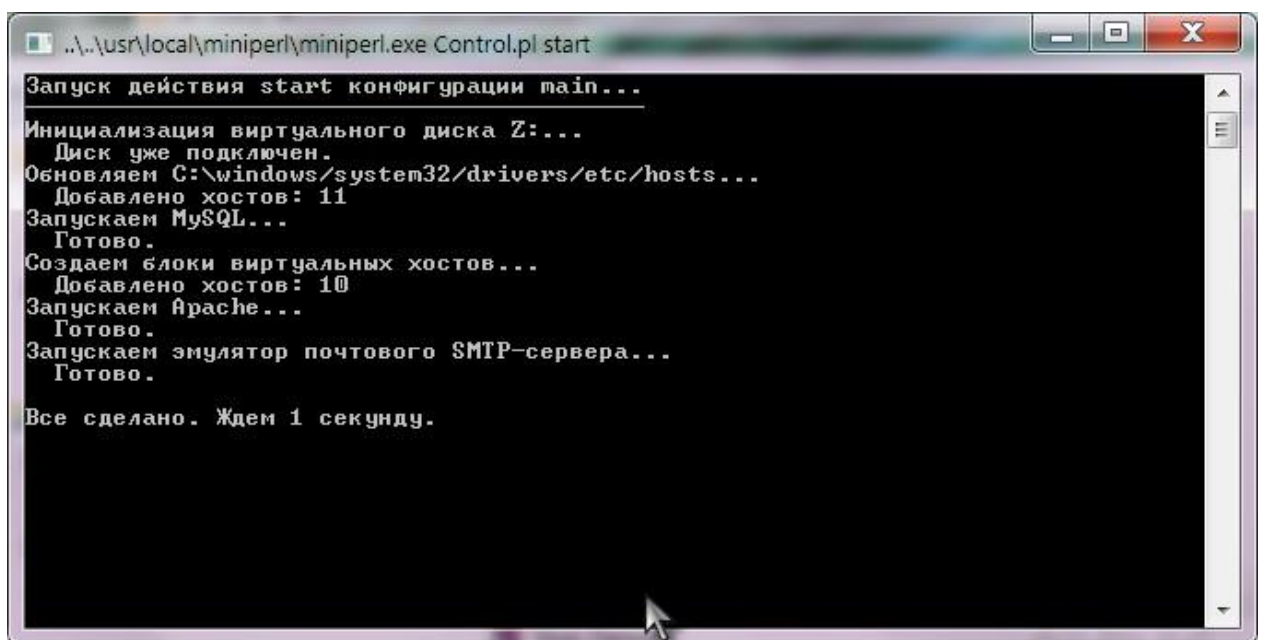


Рисунок 6.7 – Вікно ініціалізації пакету Денвер

За умови безпомилкового виконання попередніх дій введення у браузері адреси <http://localhost/> призведе до появи стартового вікна оболонки (рисунок 6.8), що свідчитиме про дієздатність WEB-серверу Apache.

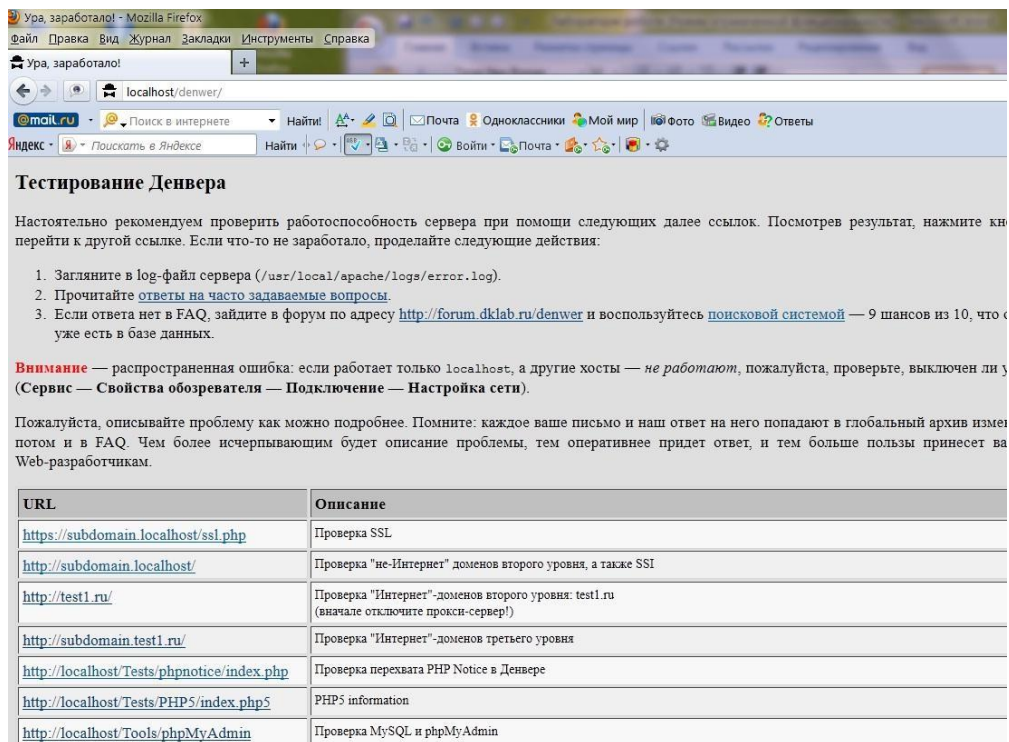


Рисунок 6.8 – Стартовое вікно пакету Денвер

5. Перевірити дієздатність MySQL-серверу та пакету phpMyAdmin. Для цього перейти у вікні браузера за адресою <http://localhost/Tools/phpMyAdmin/> та пересвідчитися (рисунок 6.9) у доступності 17-ти таблиць бази даних information_schema. Безпосередньо у браузері провести дослідження основних функціональних можливостей пакету phpMyAdmin.

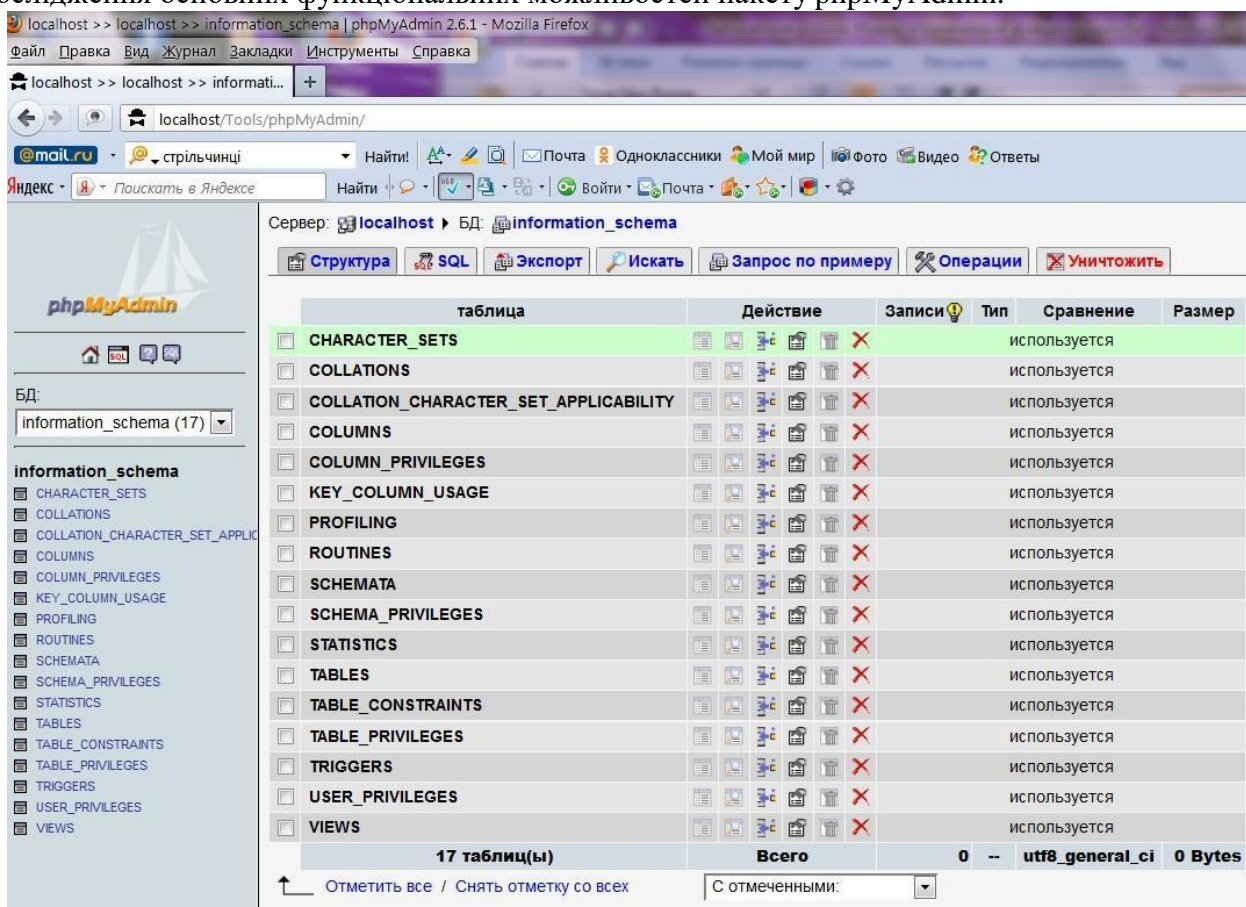


Рисунок 6.9 – База даних information_schema у пакеті phpMyAdmin

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

- a. Створити у папці Z:\home\localhost\www\ файл *create_tables.php* з та- ким змістом:

```
<?php
    include('inc.php');                                //   приєднуємо   файл   з
інформацією про підключення
    include('inc_loc.php');
    $db = mysql_connect($cServname,$cUsername,$cPassword)      //
під'єднатись до MySQL сервера
    or die('Unable to connect to MySQL server. ');           //   або   вивести   по-
відомлення про неможливість під'єднання
    if (!mysql_select_db( $cDatabase, $db ))                   //   якщо   немож-   ливо
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 - виводимо 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 - виводимо зміст помилки
    echo "<h4>Таблиця      $cTablename2      успішно      створена</h4>";
mysql_close($db);
?>
```

- b. Створити у папці Z:\home\localhost\www\ файл *inc.php* з таким зміс- том:

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

- c. 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\.

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

Вміст звіту:

1. Тема і мета лабораторної роботи.
2. У відповідності до кроків ходу роботи навести:
 - 2.1.Лістинг всіх відлагоджених програмних модулів.
 - 2.2.Результати виконання програмних модулів.
3. 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 робота №7

Тема: хостинг та FTP-доступ – основні поняття, можливості, інструменти.

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

Хостинг (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 адміністратора, 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 обслуговування 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 адміністраторі 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 завантажуватиметься 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 «Домен», де створити безкоштовне доменне ім'я. Для управління хостингом перейти на вкладку «Панель» (рисунком 7.1), де ви можете побачити створене нове доменне ім'я.

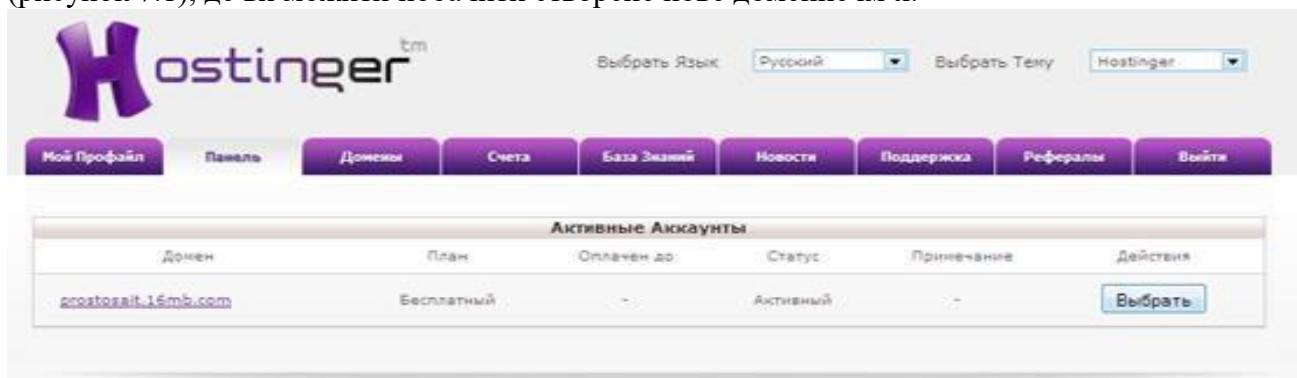


Рисунок 7.1 – Реєстрація доменного імені на hostinger.com.ua

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 7.2-7.3). Розглянути деякі з найбільш важливих служб:

1) FTP доступ – тут ви можете налаштувати параметри доступу по ftp згідно з рисунком 7.4.

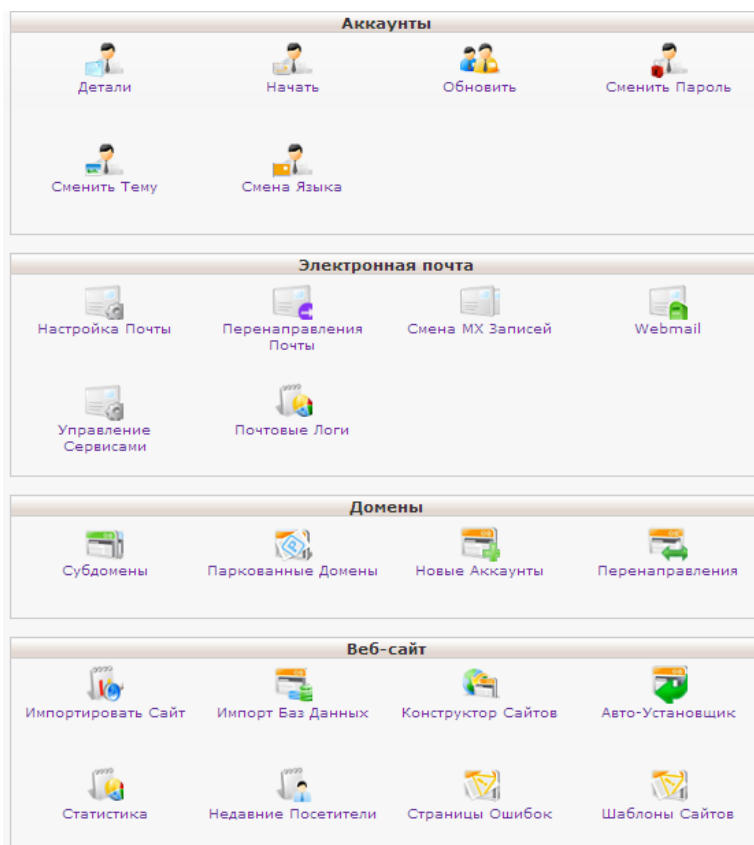


Рисунок 7.2 – Службы «Аккаунты», «Электронная почта», «Домены» та «Веб-сайт»

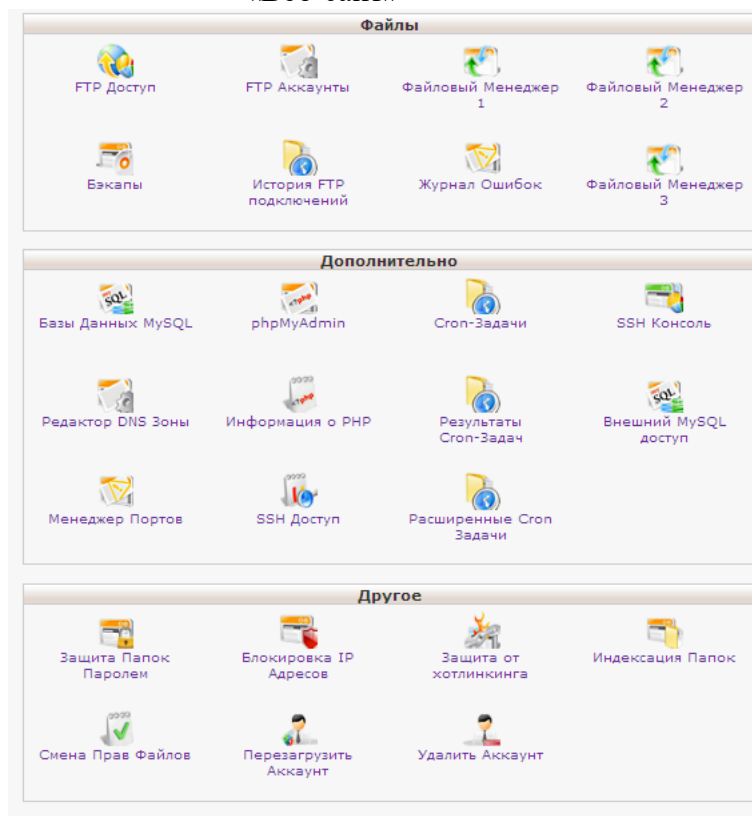


Рисунок 7.3 – Службы «Файлы», «Дополнительно» та «Другое»

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?	Сменить пароль аккаунта
Рекомендуемые FTP клиенты	SmartFTP or FileZilla

пу по ftp

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

Р
и
с
у
н
о
к
7
.
4
—
р
е
д
а
г
у
в
а
н
н
я
п
а
р
а
м
е
т
р
і
в
д
о
с
т
у

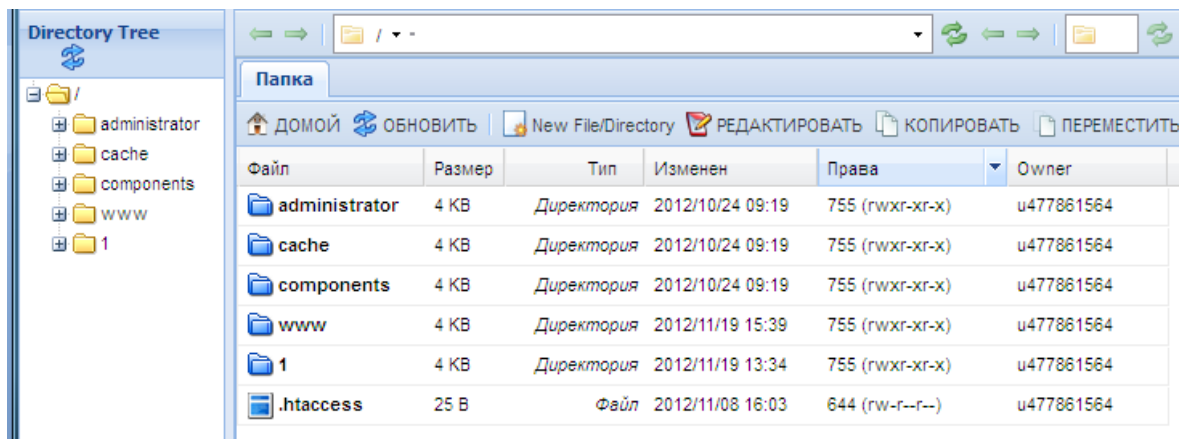


Рисунок 7.5 – служба «Менеджер файлів»

Бази Даних MySQL – служба для створення, видалення, редагування та управління доступом до баз даних MySQL (рисунок 7.6).

Создать новую MySQL базу и пользователя базы данных

Имя базы данных MySQL:

Имя пользователя MySQL:

Пароль:

Повторите пароль:

Список текущих баз данных MySQL и пользователей

База Данных MySQL	Mysql Пользователь	MySQL Сервер	Место на диске, MB	Действия
u477861564_new	u477861564_jora	mysql.hostinger.com.ua	0.02	Удалить Восстановить phpMyAdmin

Сменить пароль MySQL пользователя

Выбрать пользователя MySQL:

Пароль:

Подтвердить пароль:

Рисунок 7.6 – служба «Бази Даних MySQL»

Нарешті, охарактеризований в лабораторній роботі №1 phpMyAdmin – зручна служба для роботи з уже створеними базами даних (рисунок 7.7). Тут ви зможете повноцінно працювати з вибраною базою даних на віддаленому сервері

Список текущих баз данных		
База Данных MySQL	Пользователь MySQL	Действия
u477861564_new	u477861564_jora	Войти в phpMyAdmin

Рисунок 7.7 – 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).

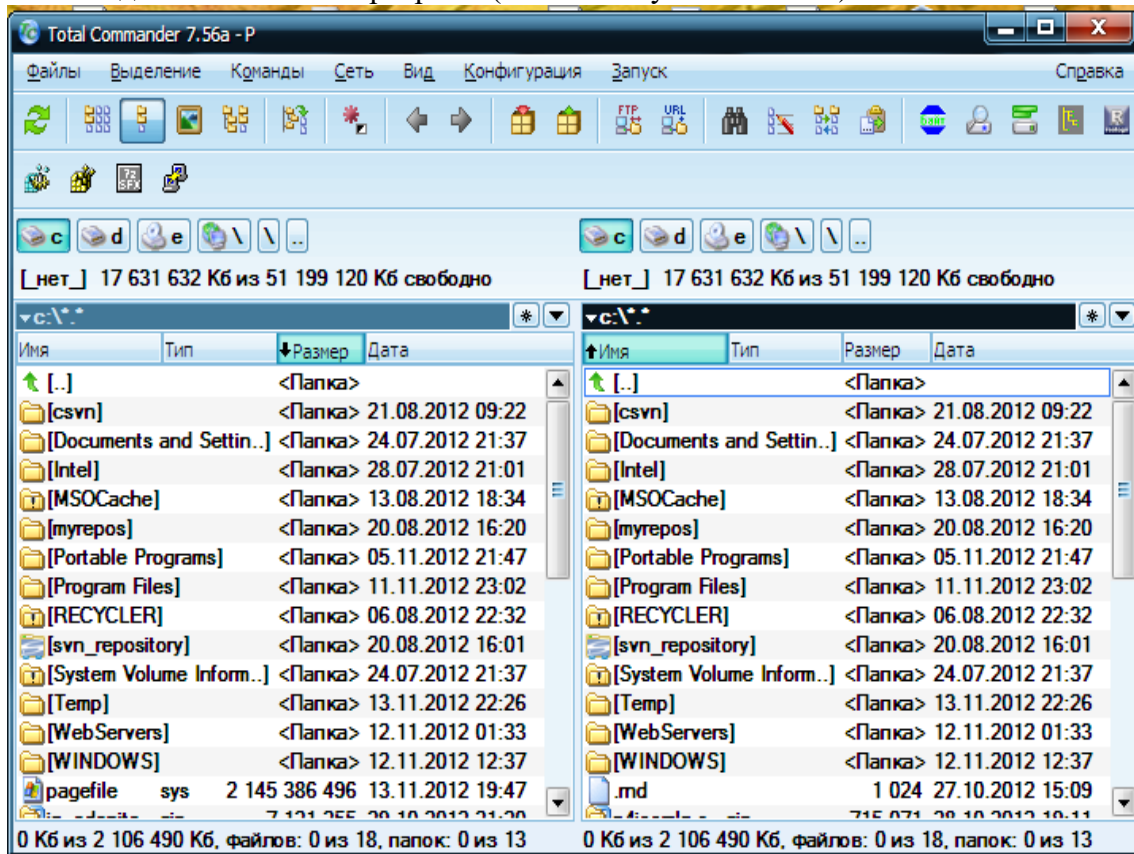


Рисунок 7.8 – налаштування FTP у Total Commander

Після цього відкриється наступне вікно вибору FTP-серверів (рисунок 7.9):

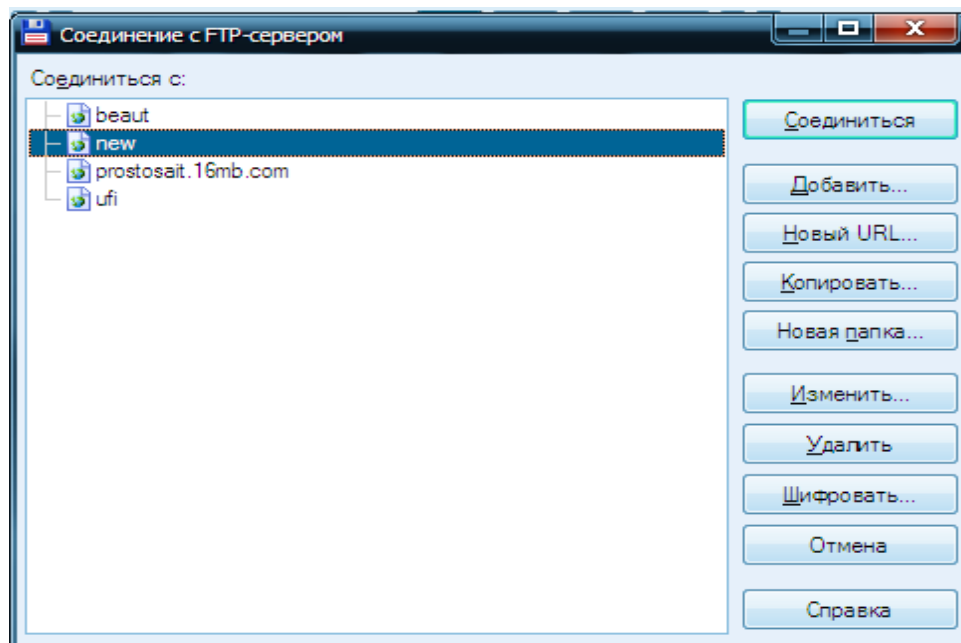


Рисунок 7.9 – вікно з'єднання з FTP-сервером у Total Commander Щоб створити новий FTP-сервер нажміть кнопку «Добавить» і побачите вікно «Настройка FTP-соединения» (рисунок 7.10):

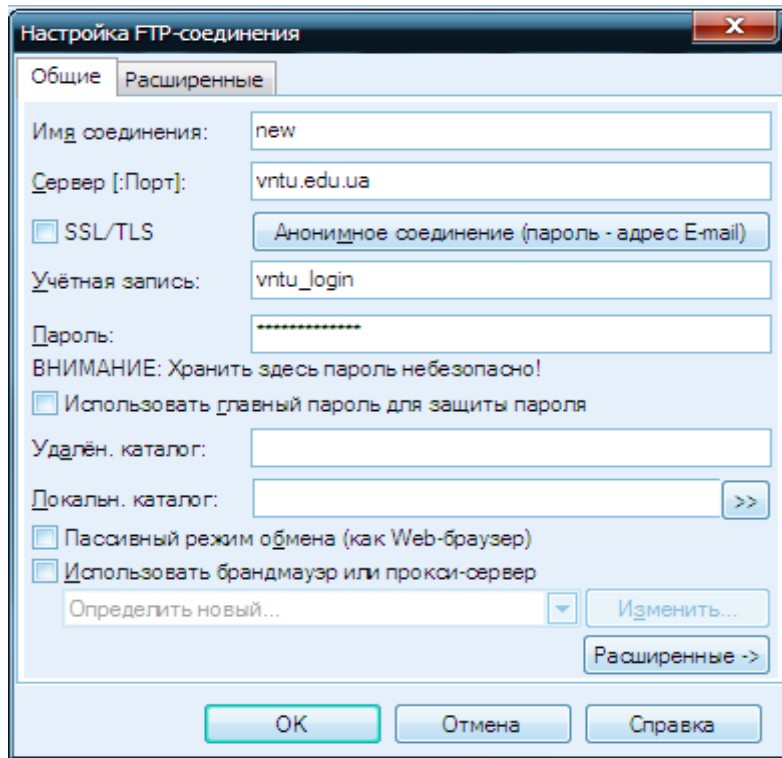


Рисунок 7.10 – вікно налаштування FTP-з'єднання у Total Commander

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

Адреса входу (address): ftp.anyhost.ru Логін

(user): ivanov

Пароль (password): 123456

Тоді адреса FTP-доступу буде виглядати наступним чином: ftp://ivanov:123456 @ ftp.anyhost.ru / (рисунок 7.13).

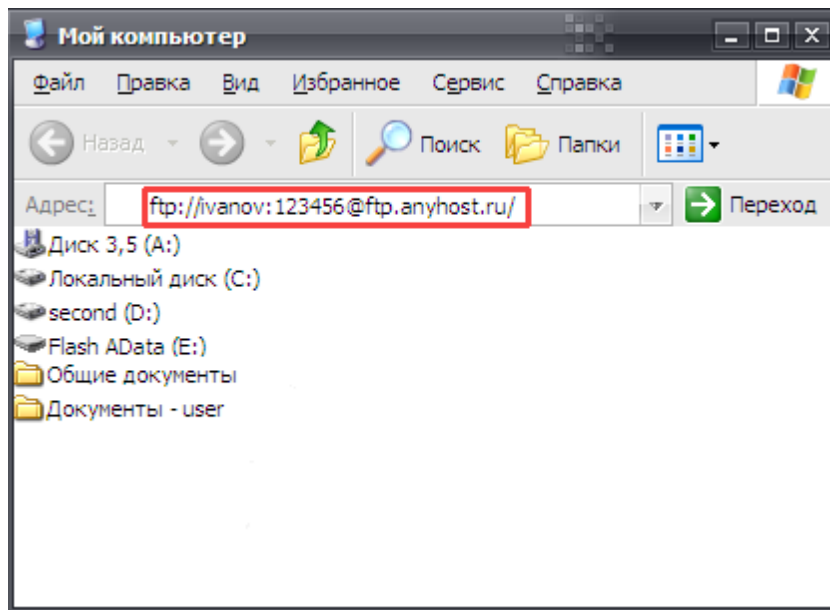


Рисунок 7.13 – створення FTP-з'єднання в режимі «Мій комп'ютер»

Після того, як Ви ввели адресу FTP-доступу, натиснути "Перехід" або кла- вішу ENTER. Якщо всі дані вказані вірно, то відкриється вміст віддаленого сер- вера у вигляді файлів і папок (рисунок 7.14).

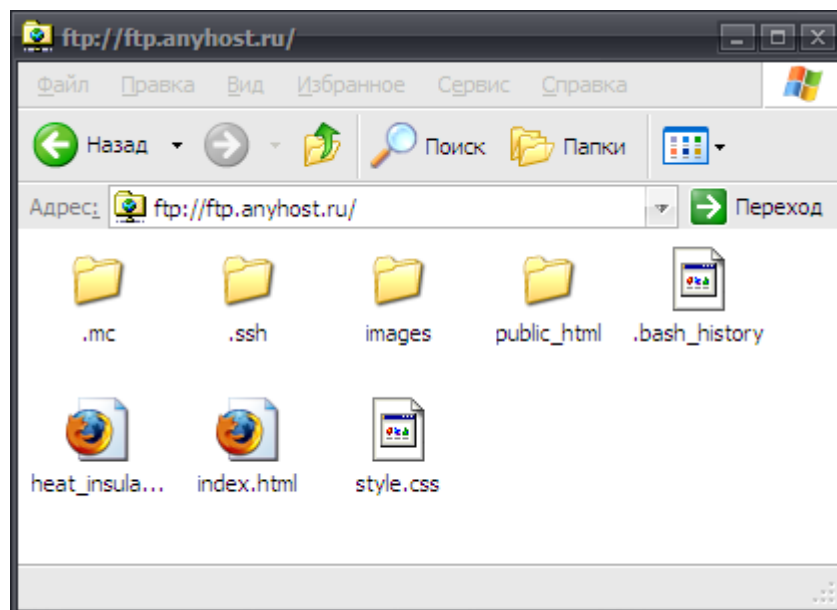


Рисунок 7.14 – вміст віддаленого сервера у режимі «Мій комп'ютер»

4.3 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 Linux 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 кон- курента.

Альтернативний метод для любителів командного рядка – ввести таку ко- манду в терміналі:

sudo apt-get install krusader

Після установки знайти Krusader можна буде в «Головному меню» – «Інші програми» – «Стандартні». На вигляд Krusader (рисунк 2.15) є звичайним двох-панельним менеджером, що дозволяє копіювати, видаляти, вирізати, перейменувати файли. Розглянемо налаштування FTP-доступу в Krusader.

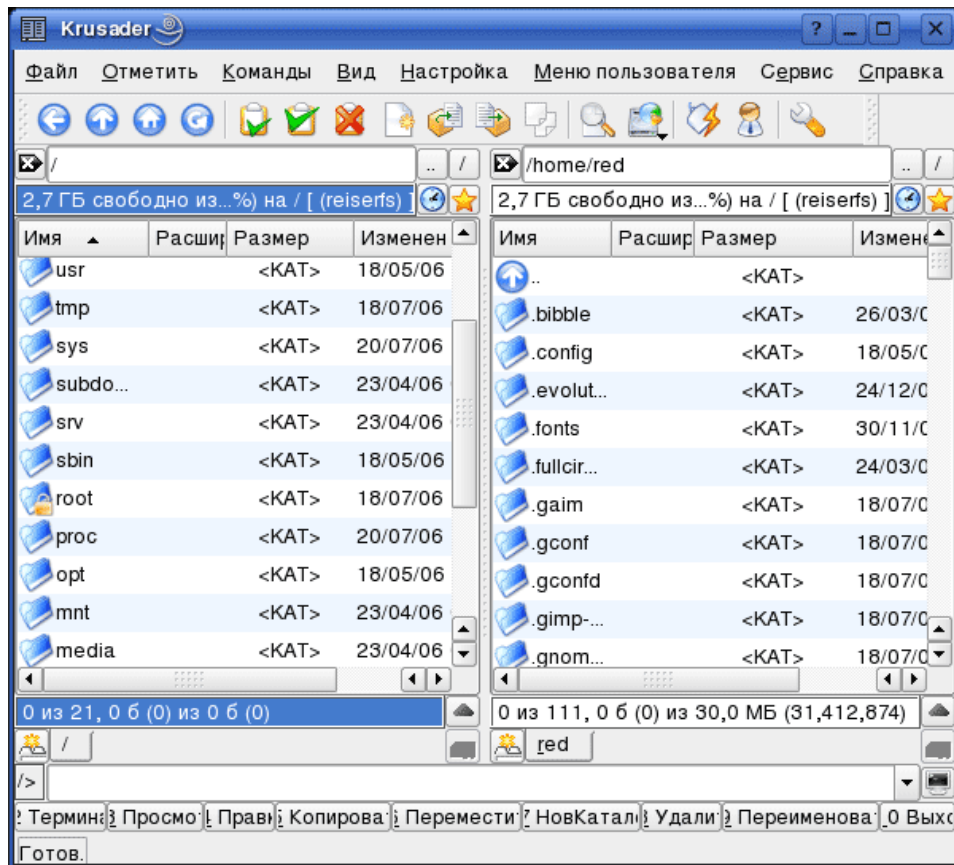


Рисунок 7.15 – зовнішній вигляд менеджера Krusader

Увійти в головне меню «Krusader»: «Сервис» – «Сетевое соединение». Відкриється нове вікно (рисунк 7.16), де необхідно послідовно заповнити поля:

Хост – адреса вашого FTP сервера;

Ім'я користувача – надається вашим провайдером ;

Пароль – надається вашим провайдером.

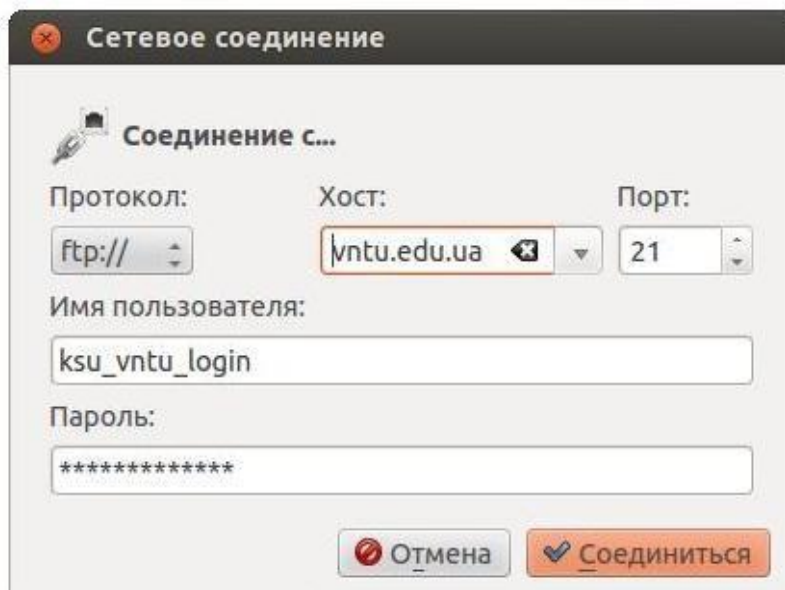


Рисунок 7.16 – створення FTP-з'єднання в Krusader

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:

1. Тема і мета лабораторної роботи.
2. У відповідності до кроків ходу роботи навести:
 - 2.1. Графічне представлення інженерних дій у вигляді схем програм, схем взаємодії програмних модулів
 - 2.2. Результати виконання всіх пунктів завдання лабораторної роботи.
3. 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 №8

Тема: основні конструкції мови сценаріїв (скриптів) PHP.

Мета: ознайомлення з основами синтаксису PHP, одержання навичок у створенні 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: браузер звертається до WEB-сервера (наприклад, my_server) із запитом файла example.php з таким кодом (наприклад, http://my_server/example.php):

```
<html>
<head>
<title> example</title>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
</head>
<body>
Приклад простої PHP-сторінки <br>
Ці стрічки не інтерпретуються, як код PHP <br> <br>
<?php
echo "Привіт, Я PHP-сценарій!";
?>
<?
echo("<p>Це теж PHP-сценарій!<p>");
?>
<?
echo("<p>Матеріал лабораторної роботи з дисципліни WEB-технології<p>");
?>
</body>
</html>
```

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 інтерпретуються, як код PHP Привіт, Я
PHP-сценарій!
Це теж PHP-сценарій!
```

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 веб-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.

Коментарі

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 (/*...*/).

```
<php
echo("<p>Hello</p>"); // коментар echo("<p>Hello</p>");
# і це також коментар
$a = "Hello, world"; echo
strstr($a,"H");
// цю функцію ми розглянемо пізніше
/*
і це теж коментарі
*/
?>
```

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. Наприклад:

```
<php
echo("<p>Hello</p>"); // правильно написаний коментар
?>
```

// а ось цей коментар буде відтворено браузером.

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 -->
```

Змінні

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.

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

?>

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

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:

```
<form name = «назва форми»  
action = «шлях до програми, що обробляє дані» method = «метод передачі  
даних»>  
поля для введення інформації ....  
</form>
```

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

Кожне поле введення інформації має атрибут NAME (наприклад, <INPUT TYPE="text" NAME="first" SIZE="4" MAXLENGTH="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?назва1=знач1&...назва2=знач2>

У випадку використання методу POST значення цих полів передаються в заголовок запиту до сервера, тобто непомітно (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 обробля- тись скриптом 2.php:

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

2.2.Зробити форму більш привабливою за допомогою стандартних HTML- тегів:

```
<html>  
<head>  
<title> Математичний приклад</title>  
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  
</head>  
<body>  
<form ...  
.....  
</form>  
</body>  
</html>
```

2.3.Зберегти файл з формою під назвою 1.php та, з метою перевірки, отримай- те у браузері аналогічний рисунок 8.1 результат.

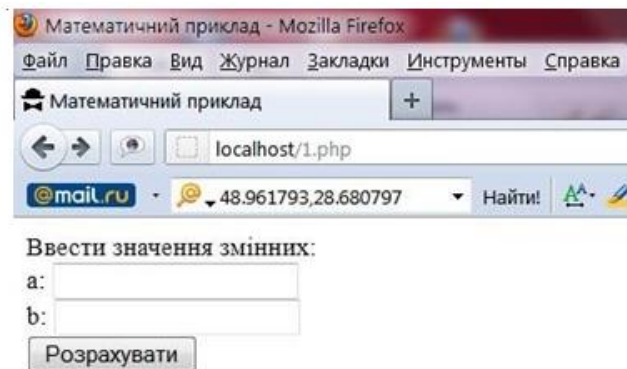


Рисунок 8.1 – Форма для введення даних у WEB-калькулятор

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 між PHP-тегами вставимо такий текст програми:

```
<?php
$string="<font color=#FF0000><b>Математичний приклад</b></font>";
$a=$_GET[a];
$b=$_GET[b];
$sum= $a+$b;
$riz= $a-$b;
$dob= $a*$b;
$dil= $a/$b;
echo "$string<br><br> "; echo "a=$a,
b=$b<br><br> ";
echo "<b>Сума </b> $sum<br> "; echo
"<b>Різниця </b> $riz<br> "; echo
"<b>Добуток </b> $dob<br> "; echo
"<b>Частка </b> $dil<br> ";
?>
```

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 розрахунків (ри- сунок 8.2). Зверніть увагу на URL в браузері – це вже робота методу GET у скрипті *2.php*.

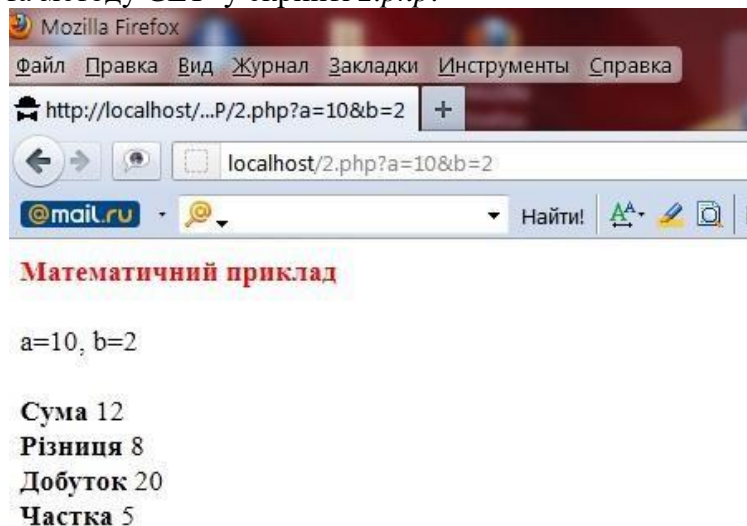


Рисунок 8.2 – Результат роботи скрипту *2.php*

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.

Вміст звіту:

1. Тема і мета лабораторної роботи.
2. У відповідності до кроків ходу роботи навести:
 - 2.1. Графічне представлення інженерних дій у вигляді схем програм, схем взаємодії програмних модулів, діаграм використання або інших UML- діаграм.

- 2.2.Лістинг всіх відлагоджених програмних модулів.
- 2.3.Результати виконання програмних модулів.
- 3. Змістовні висновки за результатами роботи.

Питання для самоконтролю:

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

Лабораторна робота №9

Тема: 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, наприклад:

<?

```
$x = 1;  
$y = 1;  
if($x==1)  
{  
    if($y==1)echo("x=1 and y=1");  
}  
else echo("x!=1");
```

?>

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**. Для об'єднання двох умов в одне застосовується операція і (**&&**). Значення змінної *\$tireqty* вводиться за допомогою html-форми (розробити таку самостійно).

<?php

```
$z="<b>Знижка дорівнює</b>";
$n="<b>Без знижки </b>";

If ( $tireqty >= 10 && $tireqty <= 49 ) {
    $discount = 5;
    echo "$z $discount<br> ";}

elseif ( $tireqty >= 50 && $tireqty <= 99 ) {
    $discount = 10;
    echo "$z $discount<br> ";}

elseif( $tireqty >= 100 ){
    $discount = 15;
    echo "$z $discount<br> "; }

                                     else

    echo "$n";
```

?>

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.

<?php

```
$names = array("Іван","Петр","Семен");  
if ($names[0]=="Іван");
```

?>

Привіт, Ваня!

<?php endif; ?>

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:

<?php if(\$HDD == «Maxtor») : ?>

```
<table>  
  <caption> Maxtor </caption>  
</table>
```

<?php elseif(\$HDD == «Seagate»): ?>

```
<table>  
  <caption> Seagate </caption>  
</table>
```

<?php endif; ?>

Результати виконання скрипта будуть такими: а) Форма запити (рисунок 9.1):

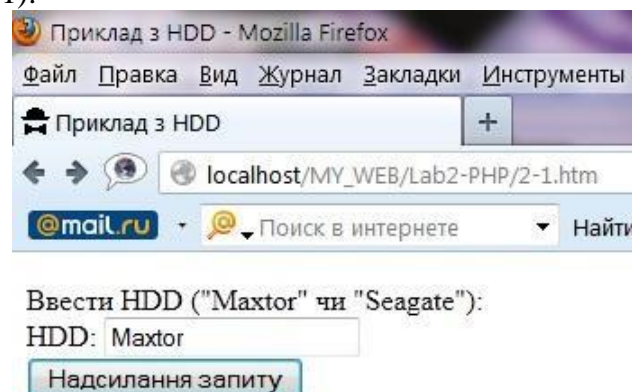
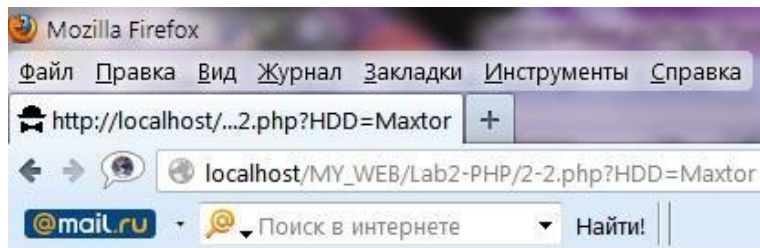


Рисунок 9.1 – вигляд форми запити 1 у браузері б) Результат виконання (рисунок 9.2):



Maxtor

Рисунок 9.2 – 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**.

Отже, наприклад, код

```
<?php
if($HDD == "Maxtor") {
    $CDROM = "Teac";
} else {
    $CDROM = "Nec";
}
?>
```

можна записати таким чином:

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

Перемикач switch

Перемикач **switch** є найзручнішим засобом для організації розгалуження із множинними варіантами умов. Синтаксис перемикача такий:

```
switch(expression) // перемикаючий вираз
{
    case value1: // константний вираз 1
        statements; // блок операторів
```

```

break;
case value2: // константний вираз 2
    statements;
    break;
default: // інакше ...
    statements;
}

```

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).

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:

```

<?
switch($number)
{
    case 1:
        echo ("one "); case 2:
    case 3: echo ("three");
    case 4: case 5: echo
("five"); case 6: case 7:
    echo ("seven"); case 8:
    case 9: echo ("nine");
    break;
    default:
        echo ("Ці номери не потрапляють у заданий діапазон > 9 or < 1");
}
?>

```

Результати виконання скрипта будуть такими: а) Форма запити (рисунок 9.3):

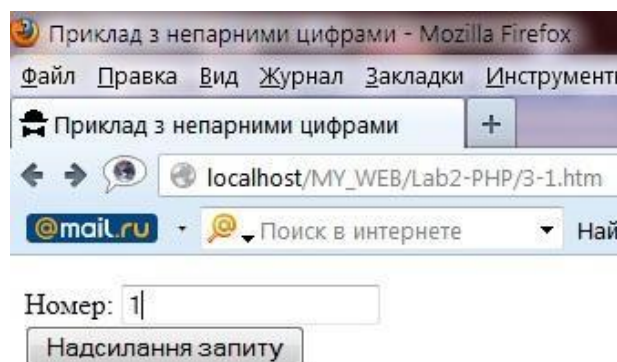


Рисунок 9.3 – вигляд форми запити 2 у браузері

б) Результат виконання (рисунок 9.4):

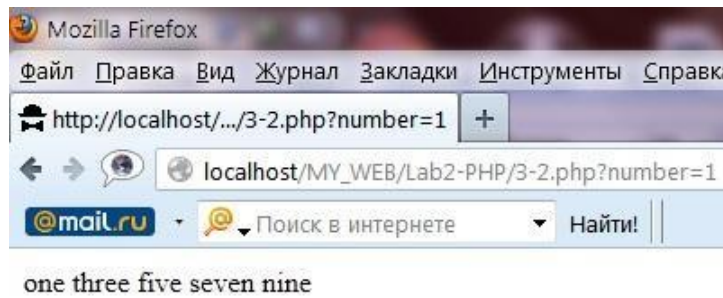


Рисунок 9.4 – результат виконання запиту 2 у браузері

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.

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;}

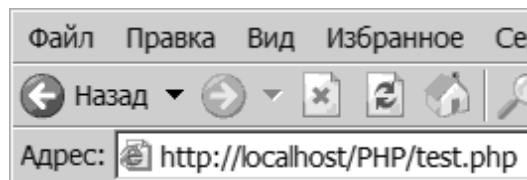
Оператор циклу / While

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. Оператором **while** зручно користуватися для проглядання довільних послідовностей, якщо в кінці їх знаходиться наперед відомий символ.

Приклад простого циклу **while**:

```
<?php
$var = 5;
$i = 0;
while(++$i <= $var)
{
    echo($i); echo('<br>');
}
?>
```

Цей код видає у вікні браузера цифри від одного до п'яти (рисунок 9.5):



1
2
3
4
5

Рисунок 9.5 – 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. При виконанні наступного прикладу, не зважаючи на те, що змінна $\$var = 7$, у вікні браузера з'являться цифри від 1 до 5.

```
<?
$var = 7;
$i = 0;
while(++$i <= $var)
{
    echo($i); echo('<br>');
    if($i==3)break;
}
?>
```

Іноді буває потрібно перервати тільки поточну ітерацію, і перейти відразу до наступної. Для цього застосовується оператор **continue**:

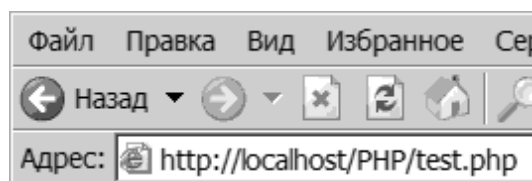
```
<?php
$var = 7;
$i = 0;
while(++$i <= $var)
{
    if($i==5)
```

```

{
    continue;
}
echo($i);
echo('<br>');
}
?>

```

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



1
2
3
4
6
7

Рисунок 9.6 – 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)
{
    ...
}

```

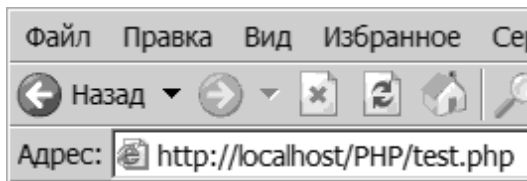
Це теж саме, що і запис **while(true)**.

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`):

```
<?php
$var = 5;
$i = 0;
do
{
    echo($i); echo('<br>');
}
while(++$i <= $var)
?>
```

Результат (рисунок 9.7):



1

2

3

4

5

Рисунок 9.7 – результат виконання прикладу для оператора **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 пред- і постумовами. За відсутності виразу-умови перед- бачається, 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:

```
<?php
$var = 5;
$i = 0;
for ($i = 0; $i <= $var; $i++)
{
    echo($i);
    echo('<br>');
}
?>
```

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 оператора **if**, яка дозволяє визначити, чи встановлено значення змінної. Наприклад, перевірка введення користувачем інформації у форму. 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.

```
If (isset ($submit)):
//Виконати якісь дії Else:
//Вивести форму Endif;
```

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

```
If (isset ($phone_nambe)):
```

```
Echo "Ви не ввели свій номер телефону!\n";  
endif;
```

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. Відповідно у 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. **Завдання №1:** відтворити приклади наведенні у лабораторній роботі.
2. **Завдання №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 (рисунк 9.8).

Будь-ласка, введіть інформацію про себе

Поля з * обов'язкові для заповнення

Ім'я	<input type="text"/>
Прізвище *	<input type="text"/>
Email адреса *	<input type="text"/>
Поштовий індекс *	<input type="text"/>
Улюблений предмет	<input type="text"/>

Готово Местная интрасеть

Рисунок 9.8 – 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 9.9).

Контактна інформація - Microsoft Internet Explorer

Файл Правка Вид Избранное Сервис Справка

Назад Поиск Избранное

Адрес: <http://localhost/php2.php> Переход

Помилка у заповненні форми!

Ви не заповнили поле Прізвище

Ви не заповнили поле Email адреса

Ім'я	Олександр
Прізвище *	<input type="text"/>
Email адреса *	<input type="text"/>
Поштовий індекс *	286021
Улюблений предмет	інформатика

Готово Местная интрасеть

Рисунок 9.9 – Результат роботи сценарію, якщо у формі були знайдені помилки

2.3. Третя функція `confirm_form`, просто відображає введену користувачем інформацію (рисунок 9.10).

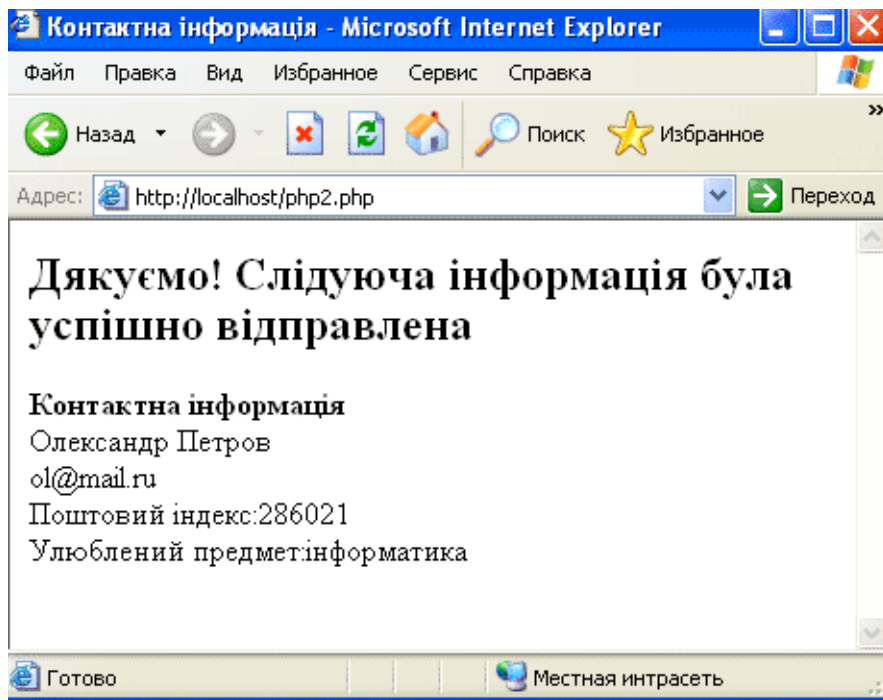


Рисунок 9.10 – Результат роботи сценарію, якщо інформація представлена правильно

Текст самого сценарію наведено нижче:

```
<html>
<head>
<title>Контактна інформація</title>
</head>
<body>
<?php
```

/ Об'являємо деякі функції*/*

```
function print_form ($f_name, $l_name, $email, $zip, $object){
    ?>
    <form action="form_checker.php" method="post">
    <table cellpadding="2" cellspacing="2" border="1">
    <tr>
        <td>Ім'я</td><td><input name="f_name" type="text" value="<?php
        echo $f_name ?>"></td>
    </tr>
    <tr>
        <td>Прізвище<b>*</b></td><td><input name="l_name" type="text"
        value="<?php print $l_name ?>"></td>
    </tr>
    <tr>
        <td>Email адреса<b>*</b></td><td><input name="email" type="text"
        value="<?php print $email ?>">
```

```

        </tr>
        <tr>
            <td>Поштовий індекс<b>*</b></td>
            <td><input name="zip" type="text" value="
<?php print $zip ?>">
        </tr>
        <tr>
            <td>Улюблений предмет</td>
            <td><input name="object" type="text" value="
                <?php print $object ?>">
            </td>
        </tr>
    </table>

    <input name="submit" type="submit" value="Надіслати">
    <input type="reset" value="Відмінити">
</form>

<?
}

function check_form ($f_name, $l_name, $email, $zip, $object){
    if (!$l_name||!$email||!$zip):echo "<h3>Помилка у заповненні фор- ми!</h3>";
    if (!$l_name){
        echo "<h3>Ви не заповнили поле <b>Прізвище</b></h3>";
    }

    if (!$email){
        echo "<h3>Ви не заповнили поле <b>Email адреса</b></h3>";
    }

    if (!$zip){
        echo "<h3>Ви не заповнили поле <b>Поштовий ін- декс</b></h3>";
    }

    print_form ($f_name, $l_name, $email, $zip, $object); else:
    confirm_form ($f_name, $l_name, $email, $zip, $object); endif;
}

function confirm_form ($f_name, $l_name, $email, $zip, $object){
    ?>
    <h2>Дякуємо! Слідуюча інформація була успішно надіслана
    </h2>
    <b>Контактна інформація</b>
    <?
    echo "<br>$f_name $l_name<br>$email<br>Поштовий ін-
    декс:$zip<br>Улюблений предмет:$object\n";
}

/* Початок основної програми*/

if (!$submit):

```



```

?>
<h3>Будь-ласка, введіть інформацію про себе</h3> Поля з <b>*</b>
обов'язкові для заповнення<p>
<?php print_form("", "", "", "", "", ""); else:
check_form($f_name, $l_name, $email, $zip, $object); endif;
?>

</body>

</html>

```

Вміст звіту:

1. Тема і мета лабораторної роботи.
2. У відповідності до кроків ходу роботи навести:
 - 2.1. Графічне представлення інженерних дій у вигляді схем програм, схем взаємодії програмних модулів, діаграм використання або інших UML-діаграм.
 - 2.2. Лістинг всіх відлагоджених програмних модулів.
 - 2.3. Результати виконання програмних модулів.
3. 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 користувацьку 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 №10

Тема: 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 табл. 10.1 позначається символом P.

Таблиця 10.1. 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.

Стрічкові типи полів наведені в таблиці 10.2.

Таблиця 10.2. Стрічкові типи

Тип	Опис
CHAR (len)[BINARY]	Стрічки 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 стрічки можуть бути довільної довжини.
TEXT	Стрічки 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 10.3.

Таблиця 10.3. 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)

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_email і ключовим полем (PRIMARY KEY) news_id:

```
mysql_query("CREATE TABLE news ( news_id INT
NOT NULL auto_increment, heading
VARCHAR(48),
body TEXT,

date DATE,

author_name VARCHAR(48), author_email
VARCHAR(48), PRIMARY
KEY(news_id))", $db);
```

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.

Зміна структури вже існуючих таблиць ALTER TABLE Команда

ALTER TABLE має три методи:

1. 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!

Робота із записами в таблицях

Додавання нового запису (INSERT)

В цій команді треба вказати назву таблиці (наприклад news), до якої потрібні вставити величини, а також вказати значення BCIX (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 обов'язково має збігатись із кількістю полів в таблиці. Вставимо, наприклад порожній запис в таблицю 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 вносити ключове число.

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

```
mysql_query("INSERT INTO news VALUES( ',  
'Назва', 'Текст  
новин', '11-16-2006',  
'Загоруйко В.',  
'sergiy@ukr.net'  
) , $db ");
```

Замість конкретних значень, можна підставляти і змінні, наприклад:

```
mysql_query("INSERT INTO news VALUES( ',  
'$a',
```

```
'Текст новин', '$dt',  
'Загоруйко В.',  
'$qwerty'  
) , $db '');
```

Зміна запис-у(ів) (UPDATE)

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,

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

Із наведеного зрозуміло, що до вказаних полів (**body** і **author_name**) маю внести відповідні значення. Але ж дещо ми забули! Тут не вказано, в ЯКОМУ записі треба зробити ці зміни? Помилка? Ні. Просто ці значення вне- суться 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:

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

Тепер зміни будуть внесені лише до запису, який має в полі `news_id` значення

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. 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 `WHERE`). Теж саме, але з умовою і відбір будемо робити за двома полями:

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

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 `*` :

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

Залишилось лише розібратись, що ж потрапляє до змінної `$x` ? Відповідь буде така: “Двовимірний масив стрічок” або просто таблиця. Тобто при відби- ранні 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, на- приклад **`body`**? Дуже просто! З індексом цього елементу в масиві. Індеси нумуються 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:

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

$y=mysql_fetch_row($x); print
"$y[0]";
```

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:

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

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:

Введення інформації в базу даних з використанням PHP

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

У основі взаємодії PHP та бази даних покладено прості принципи:

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

В даній лабораторній роботі потрібно побудувати таблицю news за допомогою PHP.

1. Створюємо файл inc_tab.php, який теж приєднується до скрипту, за допомогою функції

include:

```
<?
    $news_table = 'news';
?>
```

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:

```
<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=windows-1251">

</head>
```

1. 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.

```
<?php

include('../inc.php'); //підключаємо файл з інформацією про підключення include('inc_tab.php'); //
підключаємо файл з інформацією про таблиці include('header.php'); // підключаємо файл з
заголовочною частиною web-сторінки

$db = mysql_connect($cServname,$cUsername,$cPassword) // під'єднатись до MySQL сервера

or die('Unable to connect to MySQL server.');// або вивести повідомлення про не- можливість
під'єднання

mysql_select_db( $cDatabase, $db ) or die ("Could not activate the database");
```

// створюємо таблицю News, яка збирає замітки новин і має такі поля

```
echo"<h4>Створюємо таблицю \"$news_table\" ...</h4>";

mysql_query("CREATE TABLE $news_table ( news_id
INT NOT NULL auto_increment, heading VARCHAR(48),
body TEXT,
dt VARCHAR(10),
author_name          VARCHAR(48),
author_email         VARCHAR(48),
PRIMARY KEY(news_id))",
$db) or die(mysql_error()); //якщо запит не виконано - виводимо зміст помилки

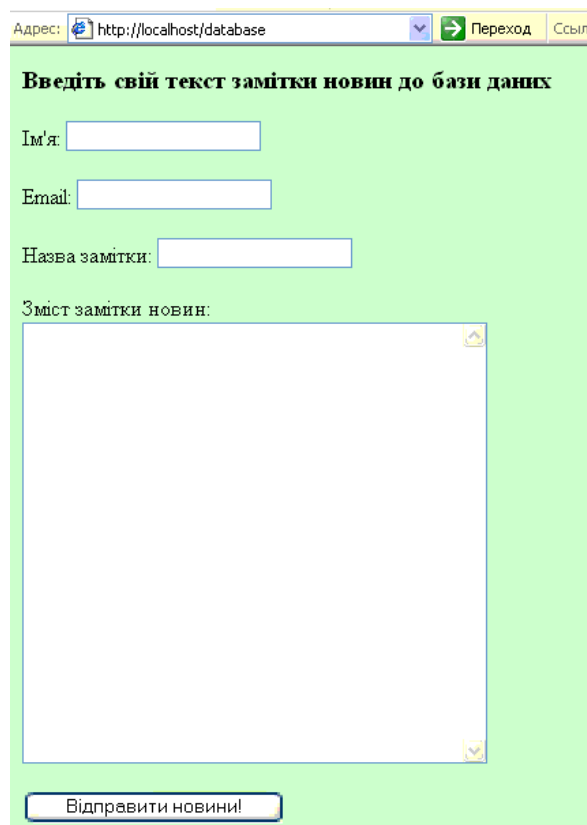
echo"<h4>Таблиці          успішно          створені</h4>";

mysql_close($db);

?>
```

1. Створюємо скрипт **database.php**, який буде заносити новини у БД.

Цей скрипт демонструє, наскільки легко розмістити інформацію в БД, ви-користовуючи 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 рис. 10.1 та 10.2



Адрес: <http://localhost/database> [Переход](#) [Ссылк](#)

Введіть свій текст замітки новин до бази даних

Ім'я:

Email:

Назва замітки:

Зміст замітки новин:

Рисунок 10.1 Початкова форма введення інформації про новини в таблицю news з скрипту **database.php**

The screenshot shows a web browser window with the address bar displaying 'http://localhost/database.php'. The page has a light green background and contains the following text and form elements:

- Запис успішно внесений.** (Record successfully entered.)
- Ви можете додати інші Новини до БД (You can add other News to the DB)
- Введіть свій текст замітки новин до бази даних** (Enter your news note text into the database)
- Ім'я:
- Email:
- Назва замітки:
- Зміст замітки новин:

Рисунок 10.2 Результат надсилення до сервера даних з форми при виконанні скрипту **database.php**

А це текст самого скрипту database.php:

```
1. <?php
2. include('../inc.php');
3. include('inc_tab.php');
4. include('header.php');
5. print"<body bgcolor='#C4FFC4'>"

6. /* Ця програма дозволяє додавати Новини у базу даних */

7. if (isset ($submit)) { //якщо була натиснута кнопка submit на формі

8. $db = mysql_connect($cServname,$cUsername,$cPassword) or die ("Не можу приєднатись до
   MySQL-сервера. Перевірте чи не порожні значення змін-них.");
9. mysql_select_db( $cDatabase, $db ) or die ("Не можу приєднатись до бази даних:
   $cDatabase.");

10.$date=date("Y-m-d");
11.if ( mysql_query("INSERT INTO $news_table VALUES ( NULL,
    '$heading',
    '$body',
    '$date',

    '$auth', '$auth_email'
  )", $db) ) {
```

```

12.echo "<h2>Запис успішно внесений.</h2>";
13.echo "<br><b>Ви можете додати інші Новини до БД</b>"; 14.} else {
15.echo "<br><b>Трапилась помилка. Перевірте скрипт.</b>"; 16.}

17.}
18.?'>

19.<p><h3>Введіть свій текст замітки новин до бази даних</h3></p>

20.<form action="database.php" method="post">

21.Ім'я: <input name="auth" type="text" ><p></p> 22.Email: <input name="auth_email"
type="text"><p></p>
23.Назва замітки: <input name="heading" type="text"><p></p> 24.Зміст замітки новин:
<br>
25.<textarea name="body" rows=20 cols=40></textarea><p></p>

26.<input type="submit" name="submit" value="Надіслати новини!">

27.</form> 28.</body>

29.</html>

```

Пояснення до скрипту database.php

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 виконуються 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 9 функція mysql_select_db обирає для подальшої роботи базу даних, назва якої знаходиться у змінній \$cDatabase.

У стрічках 11-16 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 19 за допомогою коду HTML створюється форма для введення інформації і надсилання її цьому ж самому скриптові.

Цей скрипт викликає сам себе (стрічка 26). 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.

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

3. Додайте 8-10 записів до БД.

4. Для того, щоб переглянути зміст таблиці у БД, напишіть наступний скрипт

view_news.php:

```
<?php      include('../inc.php');
include('inc_tab.php');
include('header.php');

print"<body bgcolor='#C4FFC4'>"

/* Ця програма дозволяє переглядати зміст таблиці Новин у базі даних */

$db = mysql_connect($cServname,$cUsername,$cPassword) or die ("Не можу приє-
днатись 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");

echo "<h2>Таблиця $news_table $date</h2>";

/* Тут запам'ятовуємо у змінну $result запит до серверу */

$result= mysql_query("SELECT * FROM $news_table ");

/* Змінна $items приймає перший рядок результату виконання MYSQL і розмі-
щує його в масиві
рядків. Щоб досягнути до елементів масиву використовуємо цикл while */

while ($items = mysql_fetch_row($result))

/* Виводимо результат у вигляді таблиці на екран, де у кожному комірку пропису-
ємо значення
елементу масиву за його номером */

{

echo"<table border=1> <tr align=center>
<td width=20>$items[0]</td>
<td width=60>$items[1]</td>
<td width=500>$items[2]</td>
<td width=100>$items[3]</td>
<td width=80>$items[4]</td>
<td width=80>$items[5]</td>
</tr></table>";
}
?>
</body>
</html>
```

Вміст звіту:

1. Тема і мета лабораторної роботи.
2. У відповідності до кроків ходу роботи навести:
 - 2.1. Графічне представлення інженерних дій у вигляді схем програм, схем взаємодії програмних модулів, діаграм використання або інших UML-діаграм.
 - 2.2. Лістинг всіх відлагоджених програмних модулів.
 - 2.3. Результати виконання програмних модулів.
3. Змістовні висновки за результатами роботи.

Питання для самоконтролю:

Для чого існує функція `include()`. Де і в яких сценаріях вона використовується?

Яка інформація міститься у файлі `inc.php`?

Яка інформація міститься у файлі `inc_tab.php`?

Яке призначення функції `mysql_connect()`?

Яке призначення функції `mysql_select_db()`?

Яке призначення функції `mysql_select_db()`?

Яке призначення функції `mysql_query()`?

Яке призначення функції `mysql_fetch_array()`?

Які SQL-запити ви знаєте?

10. Наведіть приклад запиту `SELECT` у зроблених вами скриптів? 11. Наведіть приклад запиту `DELETE` у зроблених вами скриптів?

12. Наведіть приклад запиту `INSERT` у зроблених вами скриптів?

13. Для чого призначена команда `UPDATE`?

14. Для чого призначена команда `DESCRIBE`?

15. Які є типи полів бази даних?

16. Як створити базу даних?

17. Як показати всі БД?

18. Як вибрати потрібну БД?

19. Як створити таблицю?

20. Що таке первинний ключ?

21. Яким чином вводяться дані у таблицю?

22. Яке призначення скрипту `database.php`? Яка послідовність створення цього скрипту? Як він працює?

23. Яке призначення скрипту `news.php`? Як він працює?

Лабораторна робота №11

Тема: методи автоматизації проектування та розробки WEB-сайтів.

Мета: ознайомлення з сучасними фреймворками, шаблоном проектування WEB-сайтів MVC (Модель–Вид–Контроллер) та CMS системами.

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:

Емблема	Назва	Короткий опис
	<u>Laravel</u>	Laravel це безкоштовний, з відкритим кодом PHP-фреймворк, створений Taylor Otwell і призначений для розробки веб-додатків відповідно до шаблону model-view-controller (MVC). Деякі з особливостей Laravel є модульна система упаковки з виділенням менеджером залежностей, різні способи для доступу до реляційних баз даних, утиліти, які допомагають в розгортанні додатків і технічного обслуговування.
	<u>Yii</u>	Високоєфективний 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).
	<u>Symfony</u>	Фреймворк з повністю відкритим кодом, написаний на PHP5. Використання цього фреймворку дозволяє створювати читабельний структурований код і багато спрощує програмування, так як величезні набори складних операцій досить часто можна замінити лише одним рядком коду.

	<u>Kohana</u>	Об'єктноорієнтований 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.
---	----------------------	--

	<p align="center"><u>CodeIgnite</u> <u>r</u></p>	<p>Всім відомий фреймворк для розробки величезної кількості веб-додатків, що використовує PHP як мову програмування. Головна його мета полягає в тому, щоб допомогти вам розробляти інтернет-проекти набагато швидше, ніж ви б змогли, вводячи весь код самому.</p>
	<p align="center"><u>Zend</u> <u>Framework</u> <u>k</u></p>	<p>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 спроектованого і написаного скрипта. Творці Zend Framework спроектували велику кількість класів, які дають можливість здійснювати різноманітні завдання, що ставляться перед командою web- програмістів.</p>

Поняття CMS

Система управління контентом сайту — це спеціалізоване програмне забезпечення, яке призначене для швидкої модифікації новин сайту та управління іншими елементами його вмісту. Така система може бути випущена в якості десктопного клієнта або у вигляді інтернет-додатки. CMS дозволяє адміністратору сайту не тільки редагувати новини в автоматичному режимі, але і створювати новий контент, додавати елементи стилістичного оформлення і користувальницького функціоналу.



Основні функції і можливості, якими володіють всі системи управління сайтом:

1. Можливість управління контентом без використання мови гіпертекстової розмітки і каскадної таблиці стилів.
2. Зберігання даних, забезпечення доступу до баз даних сайту, керування потоками документів.
3. Публікація контенту.
4. Надання зручного пошуку по вмісту сайту.

Рейтинг безкоштовних CMS

Безкоштовні версії CMS мають набагато менший функціонал, ніж платні варіанти, однак вони забезпечують основний набір функцій, який необхідний для базового управління всім вмістом вашого сайту.

Рейтинг CMS, які не мають періоду пробного використання і поширюються виключно на безкоштовній основі:

1. Data Life Engine (Malware Free). Дана система в більшості випадків використовується для новинних порталів, адже функціонал мінімальний. Однак це не заважає даній системі створювати сайти з зручною користувача навігацією. Вся документація поширюється безкоштовно. Клієнтський додаток доступно для скачування всім бажаючим створити свій блог або новинний портал. Дана CMS дає власникам сайту можливість підключити рекламу та інші системи монетизації. Підтримується функція зберігання інформації в базі даних.
2. WordPress – одна з найбільш популярних і затребуваних сьогодні систем. Також входить в рейтинг CMS з найбільш зручним розподілом функцій. WordPress вимагає наявності передвстановленого клієнта. Використовується така система, як правило, для створення складних інформаційних ресурсів, які вимагають складної організації постійної підтримки великої кількості адміністраторів.
3. Туро3. Такий варіант CMS використовується для створення корпоративних веб-сайтів. Система поширюється на безкоштовній основі з відкритим вихідним кодом. Наявність якісної документації, переведеної на різні мови, дозволяє розробникам підтримувати сайт і впроваджувати в його функціонал передові технології в області веб-програмування.



Рейтинг CMS-інтернет магазинів

1. Magento. Дана система має дуже гнучку логіку, що дозволяє задовольнити всі потреби відвідувача сайту.
2. VirtueMart. Легкий і простий движок, підійде для новачків. Система має інтуїтивно зрозумілий інтерфейс. З плюсів можна відзначити покрокову встановлення та наявність російської мови, що є рідкістю для більшості CMS.
3. WooCommerce. Доступний як в якості платної, так і у вигляді безкоштовної версії. У Росії ця система управління не користується великою популярністю, адже движок орієнтований на західний ринок і іноземного споживача.
4. PrestaShop. Цей двигун був розроблений у 2009 році. Навіть зараз він користується популярністю, адже дозволяє створити власний інтернет-магазин абсолютно безкоштовно. Така можливість чудово підійде початківцям підприємцям, які тільки пізнають всі аспекти інтернет-торгівлі.



Найбільш продуктивні CMS

Рейтинг CMS для сайту, які мають найбільшу продуктивність, виглядає наступним чином:

1. Opencart. Після персональної оптимізації система зможе витрачати мінімум ресурсів і дозволить сайту працювати набагато швидше.
2. «1С Бітрікс». Движок дозволяє управляти інтернет-магазином з більш ніж сотнею тисяч товарів різних категорій. При цьому загальна продуктивність сайту не страждає.
3. Magento. Цей движок працює стабільно. Ризик постійних зависань або збоїв системи мінімальний.

Система управління інтернет журналом

Такі варіанти движків відмінно підійдуть для початківців блогерів або для тих, хто вирішив завести власний інтернет журнал без знань в області веб-розробки.



CMS magazine рейтинг:

- 1.+Web. Відноситься до числа індивідуальних систем управління контентом сайту. Дозволяє створити індивідуальний дизайн вашого блогу.
- 2.Treegraph. Підходить для розвитку будь-яких інтернет-проектів, зокрема і блогів. Підтримує Apache і БД mySQL.
- 3.WordPress. Зручний движок, який входить в загальний рейтинг CMS.
- 4.Drupal. Дозволяє будувати абсолютно будь-які підвиди інтернет-журналів, незалежно від рівня їх складності.

Платні системи управління вмістом сайтів. Варто їх використовувати?

Перш ніж вибрати платний движок, випробуйте його безкоштовну версію протягом хоча б одного місяця. Так ви зрозумієте, чи підходить вам ця система. Платний варіант движка надає на

порядок більше функцій, наприклад, стороння реклама буде відключена, і надалі тільки адміністратор зможе підключати рекламні банери, які будуть приносити гроші власнику сайту.



Платні системи мають підвищений рівень безпеки, що є надзвичайно важливим критерієм, якщо ви плануєте створити серйозний веб-ресурс. При виборі движка керуйтеся не загальноприйнятими стереотипами, а своїм досвідом використання безкоштовної версії того чи іншого продукту.

1. Шаблиони проектування Модель – Вид – Контролер

Model-view-controller (Модель-вид-контролер) – схема використання декількох шаблонів проектування, за допомогою яких модель даних програми, користуальницький 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 архітектури каркаса, коли переходять від теорії до реалізації в конкретній предметній області. Схема концепції шаблону наведена на рисунку 8.1.

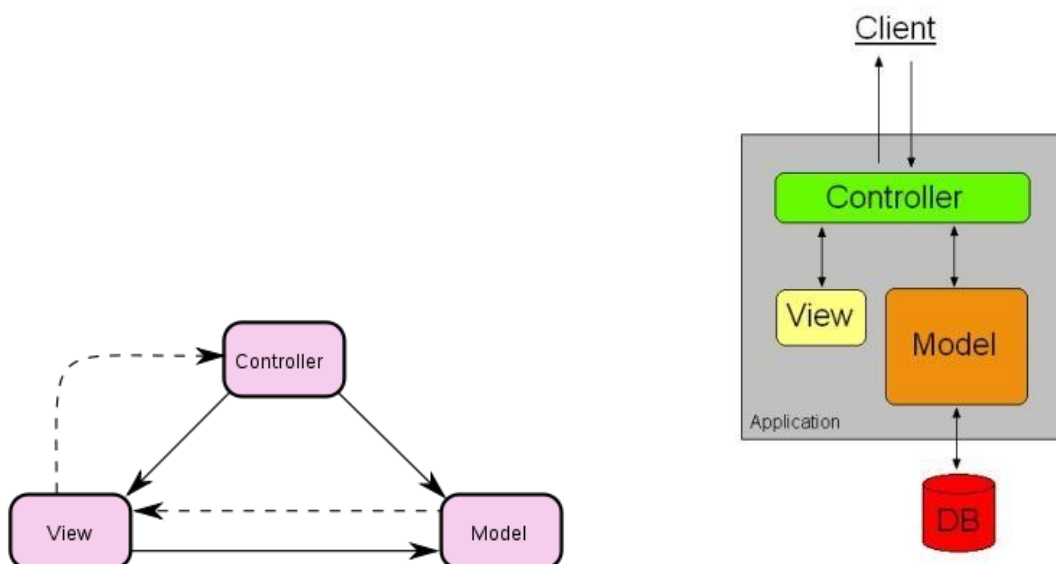


Рисунок 8.1 – Концепція Model-View-Controller. Суцільними лініями показані прямі зв'язки (виклики методів, присвоєння значень полів), переривчастими лініями показані непрямі зв'язку (повідомлення через події).

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 – модель 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. Така модель частіше використовується в структурному програмуванні, так як в цьому випадку модель являє просто структуру даних, без методів, які їх обробляють.

2. Активна модель – модель 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 кнопки, 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 добути, 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 (подій / оповіщення). При кожній зміні внутрішніх даних в моделі, модель оповіщає всіх залежних від неї представлень, і вони оновлюються. 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 підоб'єктом 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).

Завдання

1. Дослідити існуючі MVC фреймворки і CMS системи та навести їх популярність серед розробників.
2. Інсталювати MVC фреймворк або CMS систему та створити тестову сторінку для демонстрації роботи фреймворка чи CMS.

Вміст звіту:

1. Тема і мета лабораторної роботи.
2. У відповідності до кроків ходу роботи навести:
 - 2.1. Графічне представлення інженерних дій у вигляді схем програм, схем взаємодії програмних модулів, діаграм використання або інших UML-діаграм.
 - 2.2. Лістинг програмних модулів.
 - 2.3. Результати

виконання програмних модулів.

3. 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 переваги та недоліки притаманні MVC-системам?