МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний Університет Запорізька Політехніка

ГЛОБАЛЬНА ІНФОРМАЦІЙНА ІНФРАСТРУКТУРА

Методичні вказівки до лабораторних робіт
студентів спеціальності
172 Телекомунікації та радіотехніка
ОПП «Інформаційні мережі зв'язку»,
«Радіотехніка»
усіх форм навчання

2022

Глобальна інформаційна інфраструктура. Методичні вказівки до лабораторних робіт для студентів спеціальності 172 «Телекомунікації та радіотехніка», ОПП «Інформаційні мережі зв’язку» та «Радіотехніка» усіх форм навчання. – Запоріжжя: НУ «Запорізька Політехніка», 2022. – 46 с.

Укладач: Морщавка С.В., доц., к.т.н.

Рецензент: Мороз Г.В.

Відповідальний за випуск: Морщавка С.В.

Затверджено
на засіданні кафедри «Радіотехніки
та телекомунікації»
Протокол № 2 від 10 жовтня 2022

Затверджено
на засіданні НМК ФРЕТ
Протокол № 1 від 10 жовтня 2022

ЗМІСТ

ЛАБОРАТОРНА РОБОТА №1. СТВОРЕННЯ ДИНАМІЧНОГО HTML-ДОКУМЕНТА

Методичні вказівки:

Мова розмітки гіпертексту HTML (Hypertext Markup Language)

HTML (Hypertext Markup Language) - мова розмітки гіпертексту, використовуваний для створення документів, незалежних від апаратно-програмної платформи. HTML - це не мова програмування, а описова мова розмітки.

HTML-документ складається з тексту, який являє собою вміст документа, і тегів, які визначають структуру й зовнішній вигляд документа при його відображенні броузером. Структура HTML-документа проста, що видно з прикладу:

```
<html>
<head>
<title> HTML-документ</title>
</head>
<body>
<i>Internet</i>
</body>
</html>
```

Текст усього документа береться в теги <html>. Текст документа складається із заголовка й тіла, які виділяються відповідно тегами <head> і <body>. У заголовку вказують назву HTML-документа й інші параметри, які браузер буде використовувати при відображенні документа. Тіло - це та частина, у яку вкладається власне вміст HTML-документа. Тіло включає призначений для відображення текст і керуючу розмітку документа (теги), які використовуються браузером. Теги містять вказівки про спосіб відображення тексту. За допомогою тегів, крім того, створюються посилання на файли, що містять додаткові дані (графіку, звук), й позначаються точки прив’язки (гіперсилки або якоря), за допомогою яких даний документ зв’язується с іншими документами.

HTML-тег складається з імені, за яким може слідувати необов'язковий список атрибутів тегу. Текст тегу береться в кутові дужки < >.

Атрибути тегу ідуть за іменем і відділяються друг від друга одним або декількома знаками табуляції, пробілами або символами кінця рядка. Порядок запису атрибутів у тегу значення не має. Значення атрибута, якщо таке є, іде за знаком рівності, що стоїть після імені атрибута. Якщо значення атрибута - одне слово або число, то його можна просто вказати після знака рівності, не виділяючи додатково. Усі інші значення необхідно брати в лапки,

особливо якщо вони містять кілька розділених пробілами слів.

Регістр символів в іменах тегів і атрибутів не враховується.

Як правило, теги складаються з початкового й кінцевого елементів, між якими розміщуються текст і інші елементи документа. Ім'я кінцевого тегу збігається з іменем початкового, але перед іменем кінцевого тегу ставиться коса риска / (<html>. …</html>).

Кінцеві теги ніколи не містять атрибутів.

При використанні вкладених тегів їх потрібно закривати, починаючи із самого останнього й рухаючись до першого.

Деякі теги не мають кінцевого елемента.

У деяких випадках кінцеві теги можна опускати. Проте, рекомендується включати кінцеві теги, щоб уникнути помилок при відтворенні документа.

HTML надає безліч тегів, які можуть використовуватися при створенні документів. Нижче в таблиці наведені лише деякі з них.

Табиця 1 – Популярні теги

Тег	Опис
<html>…</html>	початок і кінець усього документа
<head>…</head>	початок і кінець заголовка документа
<title>…</title>	заголовок документа
<body>…</body>	початок і кінець тіла документа
…	відображення тексту курсивом
<t»...</t»	відображення тексту жирним шрифтом
<hn>…</hn> n=1…6	заголовок рівня n
…	установка шрифту, його кольору й розміру
<table>…</table>	таблиця
…. …. ….	нумерований список
…. …. ….	ненумерований список
<marquee>...</marquee>	Рядок, що біжить
<img>...</img>	вставка зображення
<a>...	гіперпосилання
<form>...</form>	форма
...	оформлення абзацу
 	переклад рядка
<hr>...</hr>	горизонтальна лінія

Каскадні таблиці стилів (CSS - Cascading Style Sheets)

Каскадні таблиці стилів - важлива частина розробки Web-додатків.

Каскадні таблиці стилів визначають макет HTML-документа у форматі, відділеному від власне інформаційного наповнення HTML-документа. Стилі можна реалізувати трьома способами:

- зв'язані таблиці стилів (Linked Style Sheets) - таблиця стилів визначається в окремому текстовому файлі з разширенням .css і її стиль зв'язується з однією або декількома сторінками. Зв'язані стилі впливають на окремий Web-вузол;
- впроваджені таблиці стилів (Global Style Sheets) - стилі можна впровадити безпосередньо в текст написаний на HTML. Впроваджені стилі впливають на окрему сторінку;
- вбудовані таблиці стилів (Inline Style Sheets) - вбудовані стилі створюються за допомогою атрибута style. Вбудовані стилі впливають на окремий тег.

Приклад використання зв'язаної таблиці стилів наведено нижче.

Таблиця стилів (у файлі lss.css)

```
h1 {font-size: 10; color:red}
p {color:#0000ff; font-style:italic}
```

Приклад HTML-документу

```
<html>
<head><link href=lss.css rel=stylesheet></head>
<body>
<h1>Заголовок нового стилю</h1>
<p>Абзац нового стилю</p>
</body>
</html>
```

Приклад застосування таблиці стилів:

```
<html>
<head><style><!h2 {font-weight:bold; color:green}></style></head>
<body><h2>Заголовок нового стилю</h2></body>
</html>
```

Приклад вбудованої таблиці стилів:

```
<html>
<h1 style="font-size:40pt; color:blue">Заголовок нового стилю</h1>
<p style="font-size:40; color: magenta">Абзац нового стилю</h1>
</html>
```

Шари

Шар – це якийсь прямокутний елемент, що містить у собі будь-яку розмітку HTML. Шаром може бути як простий рядок тексту, так і складна форма, зверстана в таблиці. За допомогою JavaScript можна змінювати розміри шару, його видимість, переміщати шар і т.п. У загальному випадку шар - це частина HTML-документа, виділена тегом div, якому привласнений деякий ідентифікатор id:

```
<div id=layer1>
</div>
```

Також необхідно, щоб шар був описаний за допомогою стильових таблиць:

```
<style type=text/css>
#layer1 {position:absolute; top=0; left=0; z-index=1;
visibility:visible; width:100px; height: 120px;}
</style>
```

За допомогою стильових таблиць описуються наступні параметри шару:

- position - крапка відліку координат положення шару, можливі значення: absolute і relative;
- top, left - координати верхнього лівого кута шару;
- z-index - рівень шару;
- visibility - видимість шару, можливі значення: visible і hidden;
- width; height - ширина й висота шару.

Приклад приховування й відображення шару:

```
<html>
<head>
<script language=JavaScript>
function showlayer(layername)
{eval('document.all[" '+layername+' "].style.visibility="visible"');}
function hidelayer(layername)
{eval('document.all[" '+layername+' "].style.visibility="hidden"');}
</script>
</head>
<body>
<style type=text/css>
#mylayer {position:absolute; top:0; left:400; z-index:1;
        visibility:visible; width:100px; height: 100px;}
```

```
</style>
<div id=mylayer>
<img src=dove.gif border=1>
</div>
<p><button onclick="showlayer('mylayer');">Показати шар</button>
<p><button oncnck="hidelayer('mylayer');">Скрити шар</button>
</body>
</html>
```

Приклад переміщення шару:

```
<html>
<head>
<script language=JavaScript>
function movelayer(layername,newtop,newleft)
{eval('document.all[" '+layername+' "].style.pixeltop=newtop');
eval('document.all[" '+layername+' "].style.pixelleft=newleft');}
</script>
</head>
<body>
<style type=text/css>
#mylayer {position:absolute; top:0; left:400; z-index:1;
          visibility:visible; width:100px; height: 100px;}
</style>
<div id=mylayer>
<img src=dove.gif border=1>
</div>
<p><button onclick="movelayer('mylayer',0,600);"> Перемістити шар
</button>
</body>
</html>
```

Об'єктна модель браузера

Об'єктна модель – це набір зв'язаних між собою об'єктів, що забезпечують доступ до вмісту сторінки й ряду функцій броузера.

Об'єкт window

Об'єкт window перебуває у вершині ієрархії і є контейнером для інших об'єктів. Він представляє поточне вікно броузера.

Методи об'єкта window

Методи, надавані об'єктом window, дозволяють управляти самим вікном, а також виконувати ряд дій усередині нього.

Таблиця 2. Властивості об'єкта window

Властивість	Опис
parent	повертає батьківське вікно для даного вікна
self	повертає посилання на поточне вікно
top	повертає посилання на саме близьке до користувача вікно
name	повертає ім'я вікна, задане тегом <frameset>
opener	повертає вікно, що створило дане вікно
closed	указує на те, що вікно закрите
status	задає текст, відображуваний у рядку стану броузера
defaultStatus	повертає текст, відображуваний у рядку стану броузера
returnValue	дозволяє події або діалоговій панелі повертати значення
document	повертає посилання на об'єкт document
event	повертає посилання на глобальний об'єкт event
history	повертає посилання на об'єкт history
location	повертає посилання на об'єкт l ocation
navigator	повертає посилання на об'єкт navigator
screen	повертає посилання на глобальний об'єкт screen

Таблиця 3 Властивості вікон

Параметр features	Значення	Опис
fullscreen	yes|no|1|0	Повноекранне або звичайне вікно (за замовчуванням звичайне)
channelmode	yes|no|1|0	Відображення смуги каналів
toolbar	yes|no|1|0	Відображення панелі інструментів
location	yes|no|1|0	Відображення адресного рядка
directories	yes|no|1|0	Відображення панелі посилань
status	yes|no|1|0	Відображення рядка стану
menubar	yes|no|1|0	Відображення рядка меню
scrollbars	yes|no|1|0	Відображення лінійок прокручування
resizeable	yes|no|1|0	Дозвіл зміни розміру вікна
width	число	ширина вікна в пікселах (min 100)
height	число	висота вікна в пікселах (min 100)
top	число	Вертикальна координата верхнього лівого кута вікна
left	число	Горизонтальна координата верхнього лівого кута вікна

Методи open і close

Для відкриття нового вікна можна скористатися методом open. Повний синтаксис методу open виглядає наступним способом:

newWnd=window.open(URL, name, features, replace),
де:
URL – адреса документа, відображуваного в новім вікні. Якщо адреса не задана, відображається порожнє вікно;
name – рядок, що задає ім'я вікна;
features – рядок, що задає параметри нового вікна;
replace – указує, чи заміщає нове вікно поточне в списку history чи ні.

Закрити вікно дозволяє метод close. Синтаксис методу close виглядає в такий спосіб:

newWnd.close()

Для закриття поточного вікна можна скористатися одним із двох способів: window.close() або self.close().

Методи alert, prompt, confirm

Ці методи дозволяють відображати різні діалогові панелі.

window. alert(«Повідомлення») виводить рядок і очікує, коли користувач клацне кнопку OK.

string=window.prompt(«Питання», «Значення за замовчуванням») служить для введення інформації. Уведена користувачем рядок вертається при клацанні на кнопці OK. При клацанні на кнопці Cancel вертається значення null.

truefalse=window.confirm(«Питання») використовується для одержання підтвердження. При клацаннях на кнопках OK і Cancel вертається true або false відповідно.

Методи focus, blur

За допомогою цих методів можна програмно переміщатися між декількома відкритими вікнами й змінювати поточне активне вікно. Метод blur переміщає фокус із одного вікна в інше (аналогічно клавіші Tab), метод focus переміщає фокус на вікно, де перебуває код, що виконується, написаний на JavaScript.

Методи setTimeout, setInterval, clearTimeout, clearInterval

Методи setTimeout, setInterval використовуються для керування таймером. Метод setTimeout створює таймер, який виконує зазначені дії після закінчення заданого числа миллисекунд, наприклад

window.setTimeout(«дія», мілісекунди).

Для виконання дій, що повторюються з певним інтервалом часу, використовується метод setInterval, наприклад

window. setInterval(«дія», інтервал у мілісекундах).

Методи clearTimeout, clearInterval скасовують дію методів setTimeout, setInterval відповідно.

Таблиця 4. Події об'єкта window

Подія	Опис
onBeforeUnload	виникає перед вивантаженням сторінки
onBlur	виникає при втраті фокуса
onError	виникає при помилці
onFocus	виникає при одержанні фокуса
onHelp	виникає при натисканні клавіші F1
onLoad	виникає в момент завантаження сторінки
onResize	виникає при зміні розмірів вікна
onScroll	виникає при прокручуванні вмісту вікна
onUnload	виникає безпосередньо перед вивантаженням сторінки

Об'єкт history

Об'єкт history містить інформацію об'єкт адресах сторінок (у форматі URL), які відвідувалися в даній сесії. Даний об'єкт має одна властивість length і три методи. Використовуючи методи об'єкта, можна переміщатися за списком history вперед та назад.

Таблиця 5. Методи об’єкту

Метод	Опис
back	завантажує попередню сторінку зі списку history
forward	завантажує наступну сторінку зі списку history
go(n)	завантажує n-у сторінку зі списку history

Об'єкт navigator

Об'єкт navigator забезпечує одержання інформації про броузер.

Таблиця 6. Властивості об’єкту

Властивість	Опис
appCodeName	кодове ім'я броузера
appName	назва броузера
appVersion	версія броузера
userAgent	частина заголовка, що посилає Web-серверу
javaEnabled	чи включена підтримка мови Java
cookieEnabled	чи включена підтримка cookies

Об'єкт screen

Для одержання інформації про клієнтський броузер використовуються значення властивостей об'єкта screen.

Таблиця 7. Властивості об’єкту

Властивість	Опис
colorDepth	максимальне число квітів, підтримуваних у даній системі
height	висота екрана в пікселах
width	ширина екрана в пікселах
pixelDepth	число біт на піксел
updateInterval	часовий інтервал відновлення екрана

Об'єкт document

Для роботи з документами HTML використовується об'єкт document. Користуючись його властивостями й методами, можна одержати інформацію про поточний документ, завантажений у вікно браузера, а також управляти відображенням змісту цього документа.

Таблиця 8. Властивості об’єкту

Властивість	Опис
alinkColor	колір активного посилання
anchors	масив локальних міток, розміщених у документі (мітки використовуються для організації посилань усередині документа)
applets	масив об'єктів, що відповідають апплетам Java, розташованим у документі
bgColor	колір тла
cookie	значення cookie для даного документа
embeds	масив plug-in об'єктів, що містяться в документі
fgColor	колір тексту
forms	масив, що містить у вигляді об'єктів усі форми, розташовані в документі
images	масив растрових зображень, включених у документ
lastModified	дата останньої зміни документа
linkColor	колір посилання
links	масив, що містить усі посилання в документі
location	повна адреса URL документа
referrer	адреса URL сторінки, що посилається на поточну
title	заголовок документа
URL	повна адреса URL документа
vlinkColor	колір посилання, що раніше відвідувався

Таблиця 9 – Методи об’єкту

Метод	Опис
open	відкриття потоку виводу в нове вікно броузера
write	вивід зазначеного тексту у вікно броузера
writeln	вивід зазначеного тексту у вікно броузера з перекладом рядка
close	закриття потоку виводу
clear	очищення вмісту обраної області
createElement	створення екземпляра об'єкта для зазначеного тегу
elementFromPoint(x ,y)	елемент, що перебуває в зазначених координатах

Таблиця 10 – Властивості об’єкту

Подія	Опис
onCllick	виникає при клацанні однієї із кнопок миші
onDblClick	виникає при подвійному клацанні однієї із кнопок миші
onMouseDown	виникає при натисканні однієї із кнопок миші
onMouseMove	виникає при переміщенні покажчика миші
onMouseOut	виникає при виводі покажчика миші з області елемента
onMouseOver	виникає при влученні покажчика миші в область елемента
onMouseUp	виникає при відпусканні раніше натиснутої кнопки миші
onDragStart	виникає при перетаскуванні елемента
onSelectStart	виникає при виборі вмісту елемента
onKeyDown	виникає при натисканні клавіші
onKeyPress	виникає при натисканні й утриманні клавіші
onKeyUp	виникає при відпусканні заздалегідь натиснутої клавіші
onKeyHelp	виникає при натисканні клавіші F1 або аналогічної для одержання довідки

Об'єкт date

Об'єкт Date і його методи використовуються для роботи з датою й часом. Дата в мові JavaScript представляється так само, як і мові Java - це число мілісекунд, що пройшли з 1 січня 1970 року. Для створення екземпляра об'єкта Date використовується конструктор new:

myDate=new Date()

Таблиця 11 – Методи об’єкту

Метод	Опис
getDate	повертає день місяця як ціле число від 1 до 31
getDay	повертає день тижня як ціле число від 0 (неділя) до 6 (субота)
getMonth	повертає номер місяця як ціле число від 0 (січень) до 11 (грудень)
getYear	повертає дві останні цифри року
getTime	повертає число мілісекунд між 1 січня 1970 року, 00:00:00 і датою, заданої об'єктом Date
getHours	повертає число годин як ціле від 0 до 23
getMinutes	повертає число хвилин як ціле від 0 до 59
getSeconds	повертає число секунд як ціле від 0 до 59
setDate	установлює день місяця
setMonth	установлює номер місяця
setYear	установлює рік
setTime	установлює час
setHours	установлює число годин
setMinutes	установлює число хвилин
SetSeconds	установлює число секунд

Використання cookie

Cookie – це механізм, що дозволяє серверу зберігати інформацію на клієнтському комп'ютері й при необхідності витягати її. За допомогою механізму cookie сервер може зберігати на клієнтському комп'ютері деякий іменований інформаційний елемент. Це може бути ім'я користувача, інформація про настроювання, службова інформація, використовувана в даній сесії й т.п. Звичайно даний механізм застосовується для збереження інформації, введеної користувачем. На якомусь вузлі користувач вводить свої дані в поля форми, вона відсилається на сервер, і інформація при цьому зберігається на комп'ютері користувача.

Механізм cookies підтримується за допомогою властивості cookie об'єкта document. Мінімально повинне бути встановлене значення атрибута name.

Таблиця 12 – Властивості об’єкту

Атрибут	Опис
name=value;	кожний інформаційний елемент зберігається у вигляді пари name=value; name задає назва елемента, value - його значення
expires=date	задає «строк придатності» інформаційного елемента: якщо цей атрибут не зазначений, строк придатності минає при закритті броузера, установка строку придатності, рівного даті в майбутньому, приводить до збереження елемента, рівного даті в минулому - видаленню елемента (дата вказується у форматі GMT)
domain= domainname	задає ім'я домена, з якого «видне» уміст даного інформаційного елемента
path=path	задає маршрут, на якім «видне» уміст даного інформаційного елемента
secure	задає захищеність інформації

Наступний приклад демонструє створення cookies.

```
<html>
<head>
<title>Cookies</title>
<script language=JavaScript>
function doCookie()
{myName="myName=";
if (document.cookie != -1)
{value=document.cookie;
alert("Hello, "+value)
}
else
{name=prompt("What is your name?", "I don't no");
document.cookie=myName+name+";";
}
} </script>
</head>
<body onload="doCookie();">
</body>
</html>
```

У наступному прикладі запитується ім'я користувача при першому відвідуванні, зберігається у вигляді інформаційного елемента, при наступних відвідуваннях відображається у вигляді вітання.

```
<html>
```

```
<head>
<title>Cookies</title>
<script language=JavaScript>
function doCookie()
{myName="myName=";
if (document.cookie.indexof(myName) != -1)
{start=document.cookie.indexof(myName);
end=document.cookie.indexof(";");
value=document.cookie.substring(start+myName.length, end);
alert("Hello, "+value)
} else
{name=prompt("What is your name?", "I don't no");
document.cookie=myName+name+";";
} }
</script>
</head>
<body onload="doCookie();">
</body>
</html>
```

Використання графіки

Керування графікою за допомогою JavaScript базується на доступі до колекції images і керуванні властивостями окремих елементів цієї колекції.

Атрибут name дозволяє звертатися до графічного зображення по імені. Наприклад, зображення, описане як <img src="First.gif" name="First">, доступно з JavaScript як document. first.

Колекція images містить усі графічні зображення, включені до складу даного HTML документа. для доступу до першого елемента можна звернутися до 0-го елементу колекції: document.images[0].

Таким чином, якщо описане вище графічне зображення було першим, можна звернутися до нього одним з наступних способів:

- document. images [0]
- document. images ["first"]
- document. first
- document["first"]

Об'єкт image

Об'єкт image може використовуватися для завдання властивостей графічних зображень, включених до складу даної сторінки, а також для завантаження зображень у кеш-пам'ять і їх наступного відображення.

У наступній таблиці перераховані властивості об'єкта image

Таблиця 13 – Властивості об’єкту

Властивість	Опис	Тільки читання	Тільки <img>
border	атрибут border тегу img	так	так
complete	булево значення, що вказує, завантажене зображення чи ні	так	ні
height	висота зображення	так	ні
hspace	атрибут hspace тегу img	так	так
lowsrc	атрибут lowsrc тегу img	ні	ні
name	атрибут name тегу img	ні	так
prototype	дозволяє додавати властивості до об'єкта image	-	ні
src	атрибут src тегу img	ні	ні
vspace	атрибут vspace тегу img	так	так
width	атрибут width тегу img	так	ні

Створення анімаційних зображень

За допомогою динамічної зміни растрових зображень у сценарії JavaScript можна одержати ефект анімації. Наприклад, це виглядає так, начебто яке-небудь слово, або малюнок, періодично тоне в кольоровому шумі, і потім проявляється знову. Вихідний текст сценарію наведений нижче.

```
<html>
<head>
<title>Animation with JavaScript</title>
<script language="JavaScript">
i=1;
bForward=true;

function showNextImage()
        {if(bForward)
                {i++;
                if(i>5)
                        {bForward=false;}
                } else
                        {i--;
                        if(i<2) {bForward=true;}
                        }
document.Img.src="noise0"+i+".gif";
setTimeout("showNextImage()",500);
```

```
}

</script>
</head>
<body bgcolor=white>
<img src="noise01.gif" name="Img">
<script language="JavaScript">
showNextImage();
</script>
</body></html>
```

Послідовність кадрів виглядає в такий спосіб:

Рисунок 1 – Ілюстрація графічного виводу

Кадри виводяться в прямій, а потім у зворотній послідовності.

Зміна зовнішнього вигляду графічних посилань

Цей прийом може бути використаний, наприклад, для зміни графічного елемента в тому випадку, коли над ним перебуває покажчик миші.

Рисунок 2 – Ілюстрація графічного виводу

Якщо розташувати покажчик миші над однією із цих кнопок, кнопка змінить свій зовнішній вигляд і з'явиться спливаюча підказка.

```
<html>
<body bgcolor="#B0FFD8">
<font face="Arial, Helvetica" size=1> <p>
<a href="back.htm" onMouseOver="document.btn1.src='back_down.gif'"
onMouseOut="document.btn1.src='back_up.gif'">
```

```
<img src="back_up.gif" name="btn1" border=0 alt="Back"></a>
<br>
<a href="forward.htm"
onMouseOver="document.btn2.src='forward_down.gif'"
onMouseOut="document.btn2.src='forward_up.gif'">
<img src="forward_up.gif" name="btn2" border=0 alt="Forward"></a>
</font>
</body>
</html>
```

Завдання

Створити HTML-документ, який повинен містити:

- форматований текст;
- багаторівневі нумеровані й ненумеровані списки;
- таблицю;
- зображення;
- гіперпосилання на інші HTML-документи, гіперпосилання в межах HTML-документа, гіперпосилання на e-mail;
- форми (input (text, checkbox, radio, submit, reset), textarea, select).
- шари;
- скрипти мовою JavaScript (відповідно до варіанта).

При форматуванні HTML-документа використовувати каскадні таблиці стилів CSS (Cascading Style Sheets): зв'язані, впроваджені й вбудовані. Продемонструвати пріоритетність CSS.

Варіант 1

Калькулятор на чотири дії (з кнопками, що натискаються). Запам'ятати в cookie результат останнього обчислення й відображати це значення на індикаторі калькулятора при повторному відвідуванні.

Варіант 2

За курсором миші переміщається деяке зображення (передбачити можливість вибору й зміни зображення). Запам'ятати в cookie ім'я файлу з останнім зображенням і відображати це зображення при повторному відвідуванні.

Варіант 3

Меню, у якому при наведенні покажчика миші пункт меню виділяється іншим кольором, при натисканні на пункті меню відкривається підменю (передбачити можливість вибору й зміни квітів меню). Настроїти й запам'ятати в cookie основний і додатковий кольори меню й використовувати

ці кольори при повторних відвідуваннях.

Варіант 4

Перегляд набору зображень зі зміною підписів до зображень за допомогою кнопок «Назад» і «Далі». При перегляді першого зображення блокується кнопка «Назад», при перегляді останнього - кнопка «Далі». Настроїти й запам'ятати в cookie шрифт для підписів до зображення й використовувати цей шрифт при повторних відвідуваннях.

Наприклад:

Фото 1. Фудзіяма.

Рисунок 3 – Ілюстрація графічного виводу

Варіант 5

Складання мозаїки. Елементи мозаїки перетаскуються за курсором миші та натисканням фіксується положення того або іншого фрагмента в мозаїці. Запам'ятати в cookie стан мозаїки й відновити його при повторному відвідуванні.

Зміст звіту:

- титульний аркуш;
- завдання;
- вихідні тексти (CSS і скрипт);
- висновки по роботі.

ЛАБОРАТОРНА РОБОТА №2. КОНФІГУРУВАННЯ Й АДМІНІСТРУВАННЯ WEB-СЕРВЕРА (НА ПРИКЛАДІ WEB-СЕРВЕРА APACHE)

Теоритичні відомості

Web-сервер Apache

Apache - один із широко використовуваних в Internet web-серверів. У цей час програмне забезпечення Apache установлене приблизно на половині Web-вузлів усього світу.

Основний файл конфігурації httpd.conf, використовуваний для керування web-вузлом. У ньому визначаються базові операції, вказується, як сервер повинен працювати з локальними ресурсами, відповідаючи на запит, вказується, з якими файлами користувачі можуть виконувати певні операції. Через нього здійснюється керування роботою Apache. Настроювання конфігураційного файлу web-сервера - самий відповідальний крок при його установці. Сервер перечитує конфігураційний файл при запуску. Якщо сервер працює, то при зміні файлу конфігурації його слід перезапустити.

У файлі httpd.conf директиви групуються в три основні розділи:

- Section 1. Global Environment - директиви, які управляють роботою Apache у цілому, впливають на загальне функціонування Apache;
- Section 2. Main Server Configuration - директиви, які визначають параметри основного сервера, що відповідає на запити, усі ці директиви можуть бути перевизначені для віртуальних серверів;
- Section 3. Virtual Hosts - установки для віртуальних серверів.

Віртуальні сервери (хости)

Віртуальні сервери (хости) - декілька web-серверів з різними IP-адресами, що використовують один екземпляр програми Apache. Сфери застосування віртуальних серверів:

- створення окремих web-серверів зі своїми адресами для різних компаній, організацій і індивідуальних користувачів;
- організація віртуальних серверів для відділів фірм, кожний відділ буде мати
  власне доменне ім'я й свій web-сервер;
- організація загальнодоступного внутрішнього web-серверів у вигляді двох віртуальних серверів;
- використання віртуального сервера для перевірки або розробки web-сервера.

Директива VirtualHost

<VirtualHost ip_address_of_host>
</VirtualHost>

Приклад: створити віртуальний сервер virthost2 с ip-адресою 127.0.0.2.

Структура каталогів виглядає в такий спосіб:

<disk:>\infocom\apache2 - каталог сервера Apache

<disk:> \infocom\virthost2\www - каталог для документів віртуального сервера

<disk:> \infocom\access.log - файл реєстрації доступу

<disk:> \infocom\error.log - файл реєстрації помилок

Section 3 у файлі httpd.conf буде мати такий вигляд:

<VirtualHost127.0.0.2>
ServerName virthost2
ServerAdmin admin@virthost2
DocumentRoot "<disk:>/infocom/virthost2/www"
CustomLog "<disk:>/infocom/access.log" common
ErrorLog "<disk:>/infocom/error.log"
</VirtualHost>

Для кожного віртуального сервера використовується своя директива DocumentRoot, тому що саме із цієї причини й створюються віртуальні сервери. Файли реєстрації доступу й помилок можуть бути тими самими.

Для організації доступу до каталогів і файлам використовуються наступні директиви.

Директива AuthType

AuthType type задає тип контролю повноважень. Можливе значення - Basic.

Директива AuthName

AuthName realm задає область (realm), у якій дійсні імена й паролі користувачів. Кожна пара ім'я/пароль діє в певній області. Можливе значення - Test.

Директива AuthGroupFile

AuthGroupFile file_name задає ім'я файлу, у якому утримується інформація про імена груп і імена користувачів, що входять у ці групи. Цей файл текстовий і має наступний формат: name_group: name_user name_user… name_group: name_user name_user…

Директива AuthUserFile

AuthUserFile file name задає ім'я файлу, у якому утримується інформація про імена користувачів їх паролях. Паролі зберігаються в зашифрованому виді. Цей файл текстовий і має наступний формат:

name_user:password
name_user: password

Приклад: імена груп, імена й паролі користувачів зберігаються відповідно у файлах <disk:>\home\security\groups і <disk:>\home\security\users. Усі імена й розташування каталогу security обрано довільно. Єдине

міркування безпеки полягає в тому, що цей каталог краще зберігати вище каталогу, зазначеного в директиві DocumentRoot.

Файл groups
adm: admin
group1: anna alex
group2: user1 user2 user3

Файл groups може бути створений за допомогою найпростішого текстового редактора. Файл users

admin:$apr1$yd5 $Bkv.cPIQk/L7EOF2d2DNO.
anna:$apr1$Ze5 $k/uC6j.ysELGdCK66BD6v0
alex:$apr1 $Gf5 $TZU pPFixB0h4pwvpf Rkxl0
user1:$apr1$Sj5 $HBg8Io5hbm/9quebHF3O01
user2:$apr1$Og5 $WKYIDjTsbk.J.PDIhXu5x1
user3:$apr1$Xi5 $AfQzJlIxyjiFh6KX23cE30

Для створення й зміни файлу users використовується Apache-утиліта <disk:>\program files\apache group\apache\bin\htpasswd.exe. Для одержання довідкової інформації її можна запустити із ключем -?. Для створення файлу використовується ключ -c. Формат команди:

\htpasswd.exe [-switch] name_user.

Для додавання другого й наступних користувачів або при зміні пароля існуючого користувача ключ -c можна не вказувати.

Після створення файлів groups і users у блокову директиву <VirtualHost> можна додати наступні рядки:

<VirtualHost127.0.0.2>
ServerName virthost2
ServerAdmin admin@virthost2
DocumentRoot "<disk:>/infocom/virthost2/www"
CustomLog "<disk:>/infocom/access.log" common
ErrorLog "<disk:>/infocom/error.log"
<Directory "<disk:>/infocom/vi rthost2/www">
AuthType Basic
AuthName Test
AuthGroupFile "<disk:>/infocom/security/groups"
AuthUserFile "<disk:>/infocom/security/users"
</Directory>
</VirtualHost>

Після створення цих файлів можна встановити права доступу до каталогів і файлам для всіх користувачів, для груп користувачів, для окремих користувачів або заборонити доступ усім користувачам.

Директива Require

- Require [user name_user name_user …] # доступ дозволений усім перерахованим користувачам
- Require [group name_group name_ group …] # доступ дозволений усім користувачам із груп
- Require [valid-user] # доступ дозволений усім користувачам перерахованим у файлі users

Приклад:

<VirtualHost127.0.0.2>
ServerName virthost2
ServerAdmin admin@virthost2
DocumentRoot "<disk:>/infocom/virthost2/www"
CustomLog "<disk:>/infocom/access.log" common
ErrorLog "<disk:>/infocom/error.log"
<Directory "<disk:>/infocom/vi rthost2/www">
AuthType Basic
AuthName Test
AuthGroupFile "<disk:>/infocom/security/groups"
AuthUserFile "<disk:>/infocom/security/users"
Require valid-user
#Require user anna user1
#Require group adm
</Directory>
</VirtualHost>

Якщо послідовно прибирати символи коментарів, то буде дозволений доступ або всім зареєстрованим користувачам, або користувачам anna та user1, або користувачам, що належать групі adm.

Директива Require дозволяє доступ користувачам на персональній основі. Для дозволу/заборони доступу всім користувачам або з конкретних ip-адрес, хостів, доменів використовуються директиви Allow from і Deny from. Ці директиви можуть мати наступні параметри: all - доступ дозволений/заборонений усім, ім'я хоста - доступ дозволений/заборонений тільки даному хосту, ім'я домена - доступ дозволений/заборонений тільки хостам з даного домена.

Порядок, у якім застосовуються директиви Allow from і Deny from, визначається не порядком їх проходження в конфігураційному файлі, а директивою Order.

За замовчуванням спочатку виконується директива Deny from, а потім

Allow from. Якщо клієнт згаданий у директиві Deny from, йому забороняється доступ за умови, що він не згаданий в Allow from. Якщо клієнт не названий у жодній із цих директив, йому доступ дозволяється.

Порядок можна змінити, використавши директиву Order allow,deny. Це значить, що доступ клієнтові, який зазначений у директиві Allow from дозволений, якщо тільки він не згаданий в Deny from. Якщо в жодній із цих директив цей клієнт не зазначений, доступ йому забороняється.

Директива Files

<Files name_file>
</Files>

Ця директива використовується для організації доступу до файлу. Установки директиви Files скасовують установки директиви <Directory>.
Приклад: каталозі www містить файл secret.html

```
<VirtualHost127.0.0.2>
ServerName virthost2
ServerAdmin admin@virthost2
DocumentRoot "<disk:>/infocom/virthost2/www"
CustomLog "<disk:>/infocom/access.log" common
ErrorLog "<disk:>/infocom/error.log"
<Directory "<disk:>/infocom/virthost2/www">
AuthType Basic
Auth Test
AuthGroupFile "<disk:>/infocom/security/groups"
AuthUserFile "<disk:>/infocom/security/users"
Require valid-user
#Require user anna user1
<Files secret.html>
Require user anna user1
#Require valid-user
</Files>
</Directory>
</VirtualHost>
```

При зміні прав доступу у файлі httpd.conf щораз потрібно перезапускати Apache. Для того, щоб уникнути перезапуску, можна помістити директиви, що визначають права доступу до каталогів і файлам безпосередньо в каталог. Для визначення імені файлу з директивами використовується директива AccessFileName.

Директива AccessFileName

AccessFileName file_name

Приклад:

AccessFileName htaccess

Якщо будуть змінюватися директиви у файлі htaccess, то це не потребує перезапуску Apache. Приклад з використання файлу htaccess:

Файл httpd.conf:

<VirtualHost127.0.0.2>
ServerName virthost2
ServerAdmin admin@virthost2
DocumentRoot "<disk:>/infocom/virthost2/www"
CustomLog "<disk:>/infocom/access.log" common
ErrorLog "<disk:>/infocom/error.log"
<Directory "<disk:>/infocom/vi rthost2/www">
AuthType Basic
Auth Test
AuthGroupFile "<disk:>/infocom/security/groups"
AuthUserFile "<disk:>/infocom/security/users"
Require valid-user
</Directory>
</VirtualHost>
Файл htaccess
<Files secret.html>
Require user anna user1
</Files>

У файл htaccess також можна поміщати й директиви із блокової директиви <Directory>, але тільки без її вказівки.

У тому випадку, якщо каталог не містить файлу index.html, Apache створює файл Index of/, який у загальному випадку виглядає не дуже добре. Для зміни зовнішнього вигляду можна використовувати поліпшену індексацію.

Директива IndexOptions

IndexOptions FancyIndexing Дана директива дозволяє застосовувати розширену індексацію.

Директива IndexIgnore

IndexIgnore file1 file2 …

Директива дозволяє виключити файли зі списку. Кілька директив IndexIgnore доповнюють одна одну.

Приклад: виключення зі списку всіх файлів з розширенням jpg і батьківського каталогу (наприклад, з міркувань безпеки).

<Directory …>

FancyIndexing on
*IndexIgnore *.jpg ..*
</Directory>

Директива AddIcon

Перший аргумент цієї директиви - ім'я файлу зі значком, другий - тип файлу, до яких значок слід додати. Якщо в якості другого аргументу використовувати значення DIRECTORY, то новий значок одержить підкаталог.

Приклад:

Alias "/icons" "<disk:>/program files/apache group/apache/icons"
*AddIcon icons/first.gif *.html*
AddIcon icons/second.gif "DIRECTORY"

Директива AddDescription

Дозволяє додати опис для певних файлів. Приклад: AddDescription "Це файли з HTML-документами" *.html

Директиви HeaderName і ReadmeName

Дозволяють формувати верхній і нижній колонтитул індексу. Ці директиви мають тільки один аргумент - ім'я файлу. Для того, щоб імена файлів, зазначених у директивах HeaderName і ReadmeName не відображалися в переліку, для них використовується директива IndexIgnore.

Завдання:

Встановити й настроїти web-сервер. Перевірити правильність настроювання. Створити два віртуальні сервери.

Розташувати кореневі каталоги документів серверів відповідно в <disk:>\infocom\virthost2\www і <disk:>\infocom\virthost3\www.

Файли реєстрації доступу й помилок розташувати в <disk:>\infocom\access.log і <disk:>\infocom\error.log. Файли з описом груп і користувачів розташувати в <disk:>\infocom\security\groups і <disk:>\infocom\security\users.

У кореневому каталозі документів одного з віртуальних серверів створити кілька каталогів і файлів. Визначити різні права доступу до різних каталогів і файлам:

- доступ дозволений усім;
- доступ дозволений окремим користувачам;
- доступ дозволено однієї групі користувачів;
- доступ дозволений усім зареєстрованим користувачам;
- доступ заборонений усім.

Перенести визначення прав доступу до одному з каталогів і одному з файлів у файл htaccess, розташований безпосередньо в каталозі, для якого визначаються права доступу.

У кореневому каталозі документів іншого віртуального сервера організувати розширену індексацію.

Зміст звіту:

- титульний аркуш;
- завдання;
- дерево створених каталогів;
- секція №3 файлу конфігурації httpd.conf;
- файли htaccess;
- висновки по роботі.

ЛАБОРАТОРНА РОБОТА №3. СТВОРЕННЯ ТЕСТОВОЇ СИСТЕМИ Й ЛІЧИЛЬНИКА ВІДВІДУВАНЬ СТОРІНКИ ЗАСОБАМИ CGI І PHP

Методичні вказівки:

CGI (Common Gateway Interface) - загальний шлюзовий інтерфейс

Один зі способів формування динамічних HTML-документів полягає у використанні додатків CGI. CGI - це інтерфейс для запуску зовнішніх програм під управлінням web-сервера.

Додаток CGI - програма, що використовує CGI-інтерфейс, одержує інформацію від вилученого користувача, обробляє її, і повертає результат (динамічно сформований HTML-документ, гіперпосилання на існуючий HTML-документ, графічне зображення і т.д.) Тому що CGI-додаток - це програма, вона повинна бути відтрансльована для тієї операційної системи, під керуванням якої працює web-сервер.

На стороні клієнта розміщається форма введення, що містить деякі поля для введення даних і кнопку для відсилання даних. Після заповнення полів і натискання кнопки дані в запиті клієнта пересилаються на сторону сервера, де web-сервер передає прислані дані CGI-додатку.

Після обробки отриманих даних CGI-додаток створює документ і передає його web-сервер, який у відповіді сервера повертає документ на сторону клієнта.

Передача інформації від клієнта до сервера й передача сформованого документа від сервера до клієнта зображена на рис. 1.

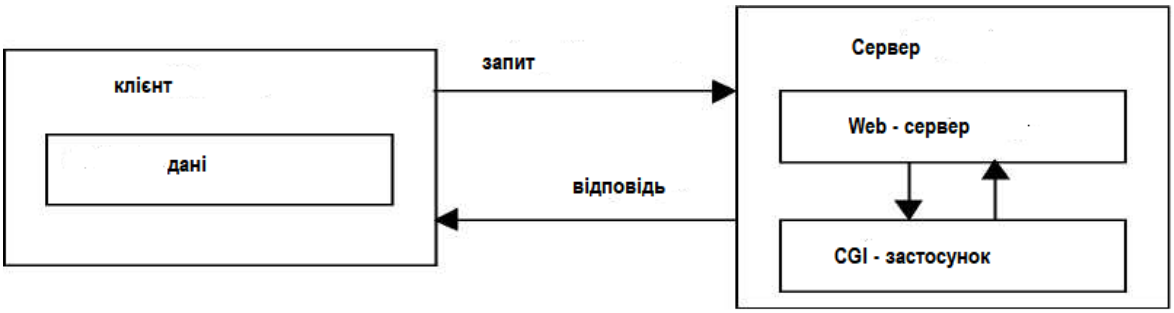

Рисунок 4 – Передача інформації від клієнта до сервера й передача сформованого документа від сервера до клієнта.

Для створення форми на стороні клієнта для занесення даних використовується тег <form>.

```
<form action=url method=get|post>
</form>
```

Атрибут action=url визначає url GCI-додатка, призначеного для обробки присланих даних. За замовчуванням використовується поточний url.

Атрибут method=get|post указує метод передачі даних серверу. За замовчуванням використовується метод get.

Метод get

Метод get припускає передачу даних GCI-додатку через змінні середовища (environment variables), установлювані на стороні сервера.

Залежно від web-сервера й операційної системи можуть використовуватися різні змінні середовища. Для передачі даних, присланих методом get, використовується змінна QUERY_STRING. Значенням змінної QUERY_STRING буде рядок, що містить дані у форматі name1=value1&name2=value2& … &namen=valuen, де name - це ім'я поля форми, value - значення поля форми.

Метод post

При використанні методу post GCI-додаток одержує прислані дані через стандартний потік уведення. Кількість байт переданих даних можна одержати через змінну середовища CONTENT_LENGTH.

Формування HTML-документа

Незалежно від методу передачі даних, результат своєї роботи GCI-додаток повинний направити в стандартний потік виводу.

Найчастіше GCI-додаток використовується для створення HTML-документів на основі даних, отриманих від клієнта. У цьому випадку, першим рядком повинен бути заголовок HTTP Content-type: text/html, за якою необхідно вивести порожній рядок, що відокремлює заголовки HTTP від даних HTML-документа.

Web-Сервер повертає результат, сформований GCI-додатком, клієнтові, можливо доповнюючи його заголовками HTTP.

GCI-додаток може сформувати повну відповідь (з усіма заголовками HTTP). У цьому випадку web-сервер нічого не змінює в результаті роботи GCI-додатка, тільки пересилає його клієнтові як є.

Приклад: на стороні клієнта в поля форми заносяться ім'я й вік, залежно від віку вертаються різні вітання (розглядаються два способи: для методів get і post).

Спосіб 1

HTML-документ, що містить форму: <html>

```
<form action=http://localhost/cgi/hello.exe method=get>
<p>ІМ'Я<input type=text name=name>
<p>ВІК<input type=text name=age>
<p><input type=submit> </form>
```

CGI-додаток (файл hello.cpp)

```
#include <iostream.h>
```

```
void main()
{
int age;
char *name;
char*query_string=getenv("QUERY_STRING");
//query_string="name=Maria&age=18"
// з рядка вилучаються підстроки "Maria" і "18"
//і привласнюються змінним name і age відповідно
cout<<"Content-type: text/html\n\n";
cout<<"<html>";
if(age<=16) cout<<"Привіт,";
if(age>16) cout<<" Здрастуйте,";
cout<<name<<"</html>";
}
```

Спосіб 2

HTML-документ, що містить форму: <html>

```
<form action=http://localhost/cgi/hello.exe method=post> <p>ІМ'Я<input type=text name=name>
<p>ВІК<input type=text name=age>
<p><input type=submit> </form>
```

CGI-додаток (файл hello.cpp)

```
#include <iostream.h> void main()
{
int age;
char *name;
intlength=atoi(getenv("CONTENT_LENGTH"));
char * string=new char[length];
scanf("%s",string);
//string="name=Maria&age=18"
// з рядка вилучаються підстроки "Maria" і "18"
//і привласнюються змінним name і age відповідно
delete string;
cout<<"Content-type: text/html\n\n";
cout<<"<html>";
if(age<=16) cout<<"Привіт,";
if(age>16) cout<<" Привіт,";
cout<<name<<"</html>";
}
```

Завдання:

У всіх варіантах завдання необхідно розробити CGI-модуль і PHP-скрипт для розв'язку одного завдання.

У всіх варіантах завдань необхідно розробити HTML-документ, що містить форму й CGI-модуль для обробки інформації, уведеної у форму, а також HTML-документ із текстом на PHP для обробки інформації, уведеної у форму.

Інформація вводиться у форму клієнтом і відсилається серверу. За результатами обробки переданої інформації динамічно генерується HTML-документ, який сервер повертає клієнтові. У вихідному HTML-документі втримується кілька питань із декількома варіантами відповіді на кожне питання. У динамічно генеруємому, що вертається HTML-документі втримується результат тестування (наприклад, оцінка). Тест складається з п'яти питань із трьома варіантами відповідей на кожне питання.

Зрівняти швидкість роботи CGI-модуля й PHP-скрипта.

Варіант 1

Перевірка знання правил дорожнього руху й текстовий лічильник відвідування сторінки.

Варіант 2

Перевірка знання таблиці множення (з генерацією співмножників датчиком випадкових чисел) і текстовий лічильник відвідування сторінки.

Варіант 3

Будь-який психологічний тест і текстовий лічильник відвідування сторінки.

Варіант 4

Перевірка знання мови HTML і текстовий лічильник відвідування сторінки.

Варіант 5

Екзамен з дисципліни й текстовий лічильник відвідування сторінки.

Зміст звіту:

- титульний аркуш;
- завдання;
- короткий опис використаних засобів і методів;
- вихідні тексти;
- висновки по роботі.

ЛАБОРАТОРНА РОБОТА №4. СТВОРЕННЯ ЕЛЕКТРОННОГО МАГАЗИНУ (ЗАСОБАМИ PHP І MYSQL)

Методичні вказівки:

Створення інтернет-магазину є складною і тривалою роботою, тому, як приклад, ми будемо створювати спрощений інтернет-магазин, який, по суті, буде лише вітриною з описом товарів, кошиком, в яку можна складати покупки, і формою замовлення. Адміністратор же зможе тільки переглядати зроблені замовлення.

Таким чином, наша база буде містити тільки дві таблиці - Товари та Замовлення. Кошик ж буде «віртуальної» і реалізовуватися за допомогою механізму Cookies.

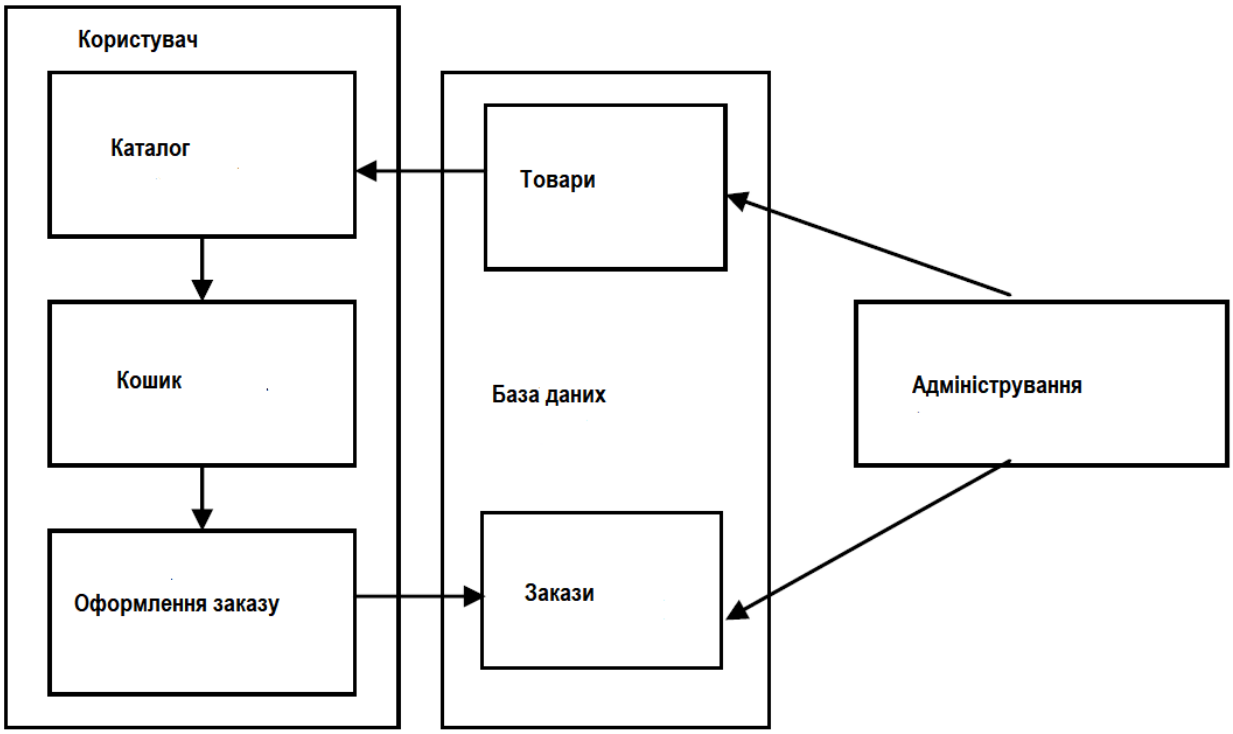

Рисунок 5 – Струтура проєкту

Для початку, розділимо процес створення магазину на кілька етапів:

1. Створення бази даних для магазину і заповнення таблиці "Товари" - файл install.php,
2. Перегляд каталогу товарів - catalogue.php (function cartStatus() і function catalogue()),
3. Кошик - вона буде складатися з кнопки оформити замовлення, очистити кошик і список товарів - cart.php,
4. Оформлення замовлення - orderform.php (html) і order.php,
5. Перегляд отриманих замовлень для адміністратора -allorders.php
6. Видалення всіх таблиць - delete. php.

Всі частини ми будемо реалізовувати у вигляді функцій.

Створення бази даних. Підготовка інформації.

Для початку роботи необхідно створити в базі даних таблиці з початковими даними. Ми заповнюватиме таблиці в явному вигляді за допомогою MySQL-запитів з PHP.

У реально існуючих інтернет-магазинах є спеціальні форми для введення, зміни та видалення даних про товар. Ми не будемо розглядати їх створення, але виконавши наступні завдання до кінця, можна подбати про таку форму. Це буде називатися інтерфейс адміністратора.

Отже, необхідно створити файл, назвемо його install.php, який, при запуску, буде створювати таблиці і вносити в них дані.

Функція №0: Створення необхідних таблиць в базі даних

```
<?
@mysql_connect("ім'я хоста","ім'я користувача","пароль") or die("MySQL Connection Failed"); /* зєднання з сервером баз даних */
@mysql_select_db("ім'я бази даних") or die("MySQL Database Selection Failed"); /* вибір бази даних */
function install() {
$content="Створення таблиць<br/>";
mysql_query("create table products(id int auto_increment primary key, name tinytext, section tinytext, description text, price float)") or die("таблиця з товарами НЕ створена<br/>");
$content=$content."таблиця з товарами створена<br/>"; /* таблиця с товарами */
mysql_query("insert into products(name, section, description, price) values('Чайник електричний Ч-23', 'чайники', 'Остання розробка компанії Tefal в області электричних чайників', '1950')") or die("товар НЕ добавлено<br/>");
$content=$content."Товар добавлено<br/>";
mysql_query("insert into products(name, section, description, price) values('Чайник електричний Ч-21', 'чайники', 'Класичний чайник з неоновим підсвітленням', '630')") or die("товар НЕ добавлено<br/>");
$content=$content."Товар добавлено<br/>";
mysql_query("insert into products(name, section, description, price) values('М ясорубка МК-415', 'м ясорубки', 'Потужна м ясорубка – рубає м ясо з кістьми!', '3400')") or die("товар НЕ добавлено<br/>");
$content=$content."Товар добавлено<br/>";
/* добавлення нових товарів */
mysql_query("create table purchases(id int auto_increment primary key, date tinytext, name tinytext, address tinytext, email tinytext, cart tinytext)")
```

```
or die("таблиця з покупками НЕ створена<br/>");
$content=$content."таблица з покупками створена<br/>";
/* таблиця з покупками */
return $content;
/* функция повертає результат створення таблиць и записів в ній*/
}
echo install();/* виводимо результат работи функції на екран */
?>
```

Каталог

Ми створили в базі даних таблицю з товарами. На наступному етапі потрібно переглянути уведені дані, тобто створити каталог товарів. Для цього використовуємо вже знайомий нам метод. Перш за все, за допомогою функції mysql_query, отримуємо в змінну $result всі дані з таблиці products. За допомогою циклу for перебираємо всі рядки з даними, де отримуємо їх у вигляді масиву за допомогою функції mysql_fetch_array.

Створимо файл с каталогом товарів catalogue.php.

Функція №1: Каталог товарів

```
<?
@mysql_connect("ім я хоста","ім я користувача","пароль") or die("MySQL Connection Failed"); /* з єднання з сервером баз даних */
@mysql_select_db("ім я бази даних") or die("MySQL Database Selection Failed"); /* вибір бази даних */
function catalogue() {
$result=mysql_query("select * from products"); /* отримуємо в $result всі товари */
$content="";
/* створюємо змінну $content, в яку будемо записувати все те, що треба показати користувачу (створення HTML-файла) */
for ($i=0; $i<mysql_num_rows($result); $i++) {
/* перебираемо всі строки в таблиці с товарами, користуючися  for */
$row=mysql_fetch_array($result);
/* отримуємо масив з даними одного рядка – окремого товару */
$content=$content."--------------------<br/><br/>
<strong>".$row['name']."</strong><br/>".$row['description']."<br/><br/>
<strong>цена:</strong> ".$row['price']." грн.<br/><br/>
<a href=\"catalogue.php?addtocart=".$row['id']."\">положить в корзину</a><br/><br/>"; /* складаємо інформацію про товар */
}
```

```
return $content; /* повертаємо змінну $content */
}
echo catalogue(); /* втведемо результат работи функції на екран */
echo "<a href=\"cart.php\">перейти в кошик</a>"; /* «до кошику» */?>
```

Додавання товару в кошик

Вся інформація про зміст кошика є тимчасовою і буде зберігатися на комп'ютері користувача до закриття вікна браузера. Для цього зазвичай використовують cookies. Cookies - це змінні зберігаються на комп'ютері користувача в форматі ім'я = значення, які доступні тільки для сайту з якого вони були записані. Щоб записати дані в cookies, в PHP використовують функцію setCookie(name,value) з двома параметрами - ім'я змінної та її значення. Ми назовемо нашу змінну cart, а її значенням буде зміст кошика у вигляді ідентифікаторів товарів, зазначених через розділовий знак (нехай це буде знак |). Щоб отримати дані з кошика, використовують стандартний масив $_COOKIE, робота з ним нічим не відрізняється від роботи зі стандартними масивами $_POST і $_GET.

Зазвичай, щоб додати товар в корзину використовують функції, написані на javascript (з якого можна теж змінювати cookies). Але вивчення javascript не входить в програму даного курсу, тому ми будемо здійснювати це іншим способом, яким в реальному житті, дуже рідко користуються, але він дозволяє зрозуміти принцип вирішення питання.

Щоб зрозуміти, як працює cookies, виконаємо маленьку вправу в окремому файлі (наприклад, prim_cart.php).

```
<?
setCookie("cart","1|2|4|5");
echo $_COOKIE["cart"]."<br>";
?>
```

Тут цифри - це будуть ідентифікатори товарів, а роздільник між ними - символ "|". Тепер спробуємо розділити їх на елементи масиву, а заодно і підрахуємо їх кількість. Продовжимо наш приклад:

```
<?
setCookie("cart","1|2|4|5");
echo $_COOKIE["cart"]."<br>";
$cartArr=explode("|",$_COOKIE["cart"]);
foreach($cartArr as $temp)
{
```

```
echo $temp." ";
}
$k=count($cartArr);
echo " В кошику $k товарів";
?>
```

Таким чином ми будемо аналізувати склад кошика.

Опція «покласти в кошик» буде являти собою посилання на цей же php-файл з ідентифікатором товару, переданим методом GET, наприклад, href = "cataloge.php? Addtocart = 1". У самому файлі, перед функцією виведення каталогу, потрібно вставити функцію перевірки на наявність змінної addtocart в адресному рядку. Крім цього, нам потрібно вивести на сторінку кількість товарів в кошику. Ці завдання ми будемо вирішувати за допомогою функції cartStatus(). Її потрібно помістити в файл з каталогом перед описом функції catalogue(). Це має виглядати наступним чином:

```
<?
@mysql_connect("ім'я хоста","ім'я пользователя","пароль") or die("MySQL Connection Failed");
@mysql_select_db("ім я базы даних") or die("MySQL Database Selection Failed");
function cartStatus() {
.....
}
function catalogue() {
.....
}
echo catalogue();
?>
```

Функция №2: Додавання товару та стан кошику

```
function cartStatus() {
/* частина перша – додавання товару в кошик */
$cart="";
/* створюємо змінну $cart, в яку запишемо кошик */
if (@$_COOKIE['cart']) { $cart=$_COOKIE['cart']; }
/*якщо існує змінна «cart» у cookies, то надаємо її значення змінній $cart*/
if (@$_GET['addtocart']) {
```

```
/* якщо існує змінна addtocart в адресній рядку (метод GET), то */
if (!@$_COOKIE['cart']) {
/* якщо не існує змінної «cart» у cookies */
$cart=$_GET['addtocart'];
/* надамо змінній $cart ідентифікатор товару, що додається в кошик */
}
else {
/* якщо існує змінна «cart» у cookies, потрібно поставити розділовий знак між ідентифікаторами (хай це буде знак | ), і тільки потім додати новий ідентифікатор */
$cart=$cart."|". $_GET['addtocart'];
}
setCookie("cart", "$cart");
/* замінимо змінну «cart» у cookies на $cart з новим, зміненим станом кошика*/
}
/* частина друга – отримання кількості товарів у кошику */
if ($cart=="") { $cartCount="0"; }
/* якщо змінна $cart, отримана раніше з cookies, є порожнім рядком, значить, кошик порожній і змінною, що містить кількість товарів, назвемо її $cartCount потрібно присвоїти значення 0 */
else {
/* якщо змінна $cart не порожня та в кошику знаходяться товари… */
$cartArr=explode("|",$cart);
/* … Потрібно створити масив з ідентифікаторами товарів. Для цього використовуємо стандартну функцію explode(). Її робота полягає у розбитті рядка на підрядки відповідно до зазначеного роздільника та розміщення отриманих даних у масив. Отже, товарів у кошику стільки, скільки елементів у масиві */
$cartCount=count($cartArr);
/* стандартна функція count() повертає кількість елементів у вказаному в дужках масиві; надамо це значення змінної $cartCount */
}
return $cartCount; /* функція повертає кількість товарів у кошику */
}
```

Функція написана, покладена в потрібне місце в файлі, але не запущена. Можна запустити її простим викликом cartStatus (), але тоді вона буде тільки додавати товар в корзину, а вона вміє ще й повертати кількість товарів в кошику. Тому будемо запускати її в такий спосіб:

```
<?
@mysql_connect("ім'я хоста","ім'я користувача","пароль") or die("MySQL Connection Failed");
@mysql_select_db("ім'я бази даних") or die("MySQL Database Selection Failed");
function cartStatus() {
.....
}
function catalogue() {
.....
}
$cartCount=cartStatus();
/* запускаємо функцію і повертаємо її значення в змінну $cartCount */
echo "Кількість товарів у кошику: ".$cartCount." шт.<br/><br/><br/>";
/* тепер у самому верху сторінки з каталогом буде вказано кількість товарів у кошику на даний момент */
echo catalogue();
/* виводимо каталог товарів */
?>
```

Кошик

Наш кошик, як і сам магазин, буде максимально простим. Вона складатиметься з кнопки оформити замовлення, очистити кошик та списку товарів без можливості зміни кількості та видалення певного товару, т.к. це є досить непростим завданням для програміста-початківця.

Таким чином, нашим завданням у цій лабораторній роботі буде отримати з cookies ідентифікатори товарів, які відвідувач поклав у кошик, знайти за ними відповідні товари в базі даних, вивести їх назви та ціни та порахувати загальну суму покупки.

Створимо файл cart.php.

Функция №3: Кошик

```
<?
@mysql_connect("ім'я хоста","ім'я користувача","пароль") or die("MySQL Connection Failed");
@mysql_select_db("ім'я бази даних") or die("MySQL Database Selection Failed");
function cart() {
$cart="";
/* створюємо змінну $cart, в яку запишемо кошик */
if (@$_COOKIE['cart']) { $cart=$_COOKIE['cart']; }
```

```
/* якщо існує змінна «cart» у cookies, то надамо її значення змінної $cart */
$cartArr=explode("|",$cart);
/* використовуємо вже відому нам функцію explode, щоб отримати масив з ідентифікаторами товарів у кошику */
if ($cart=="") {
return "кошик порожній";
/* якщо змінна $cart є порожнім рядком, то результатом функції буде напис «кошик порожній» */
}
else {
$cartCount=count($cartArr);
/* запишемо в змінну $cartCount кількість товарів у кошику, отриману за допомогою функції count() */
$cartTotalPrice=0;
/* створюємо змінну $cartTotalPrice, в якій вважатимемо загальну вартість */
foreach($cartArr as $k=>$v) {
/* за допомогою оператора foreach перебираємо всі елементи масиву, де $k – номер елемента масиву (починаючи з нуля), $v – ідентифікатор товару */
$result=mysql_query("select * from products where id='$v'");
/* отримуємо в змінну $result результат mysql-запиту «вибрати всі таблиці products, де ідентифікатор ($id) дорівнює змінній $v (ідентифікатор з масиву з товарами)» */
$row=mysql_fetch_array($result);
/* перетворимо отримані дані на асоціативний масив $row */
$number=$k+1;
/* у кожного товару в кошику має бути номер починаючи з одиниці, а php веде рахунок з нуля, тому до порядкового номера виданого php потрібно додати одиницю. Запишемо отримане значення змінну $number */
$content=$content.$number." - ".$row['name']." - ".$row['price']." руб.<br/>";
/* оформляємо отримані дані */
$cartTotalPrice=$cartTotalPrice+$row['price'];
/* додаємо до змінної $cartTotalPrice (загальна сума купівлі) ціну чергового товару */
}
$content=$content."<br/><br/> Усього: ".$cartTotalPrice." руб. ";
/* закінчили перебір товарів у кошику та виводимо загальну вартість покупки */
$content=$content."<br/><br/><a  href=\"orderform.php\">оформити замовлення</a>\n";
```

```
/* опція «оформити замовлення» */
}
return $content;
/* виведення результатів */
}
echo cart();
/* виведемо результат роботи функції на екран */
?>
```

Для очищення кошика можна використовувати функцію setCookie (), в якій змінної cart присвоюємо порожній рядок: setCookie (cart, "")

Оформлення замовлення

Оформлення замовлення буде складатися з двох частин: форма для введення даних про користувача і запис цієї інформації в базу даних разом з утримання кошика. Форму створіть самостійно в файлі orderform.php. У тезі form вкажіть method = "post" і action = "order.php". У формі повинні бути наступні поля: name (ПІБ), address (поштова адреса), email (електронна пошта).

Після написання файлу з формою, створіть файл order.php, в який ми помістимо функцію оформлення замовлення.

Функція №4: Оформлення замовлення

```
<?
@mysql_connect("ім'я хоста","ім'я користувача","пароль") or die("MySQL Connection Failed");
@mysql_select_db("ім'я бази даних") or die("MySQL Database Selection Failed");
function order() {
$name=@$_POST['name'];
$address=@$_POST['address'];
$email=@$_POST['email'];
/* у змінні $name, $address та $email отримаємо передані методом post змінні з форми */
$cart="";
/* створюємо змінну $cart, в яку запишемо кошик */
if (@$_COOKIE['cart']) { $cart=$_COOKIE['cart']; }
/* якщо існує змінна «cart» у cookies, то надамо її значення змінної $cart */
$date=time();
```

```
/* запишемо в змінну $date поточну дату */
mysql_query("insert in purchases(date, name, address, email, cart) values('$date', '$name', '$address', '$email', '$cart')");
/* додамо до таблиці з покупками новий запис */
return "Ваше замовлення надійшло на сервер. Дякую";
}
echo order();
/* виведемо результат роботи функції на екран */
?>
```

Робота зі заказами

Метою даної лабораторної роботи є отримання сторінки з усіма замовленнями. Для цього потрібно взяти з таблиці purchases всі записи і за допомогою вже добре відомого методу (через цикл for і функцію mysql_fetch_array()), вивести їх на сторінку. Завдання ускладнюється тим, що в стовпці cart знаходяться ідентифікатори товарів і їх потрібно перетворити в назви, тому що фраза «покупець замовив товар 5 і 8» Вам нічого не скаже.

Створіть файл allorders.php і помістіть в нього наступну функцію.

Функція №5: Оформлення заказу

```
<?
@mysql_connect("ім'я хоста","ім'я користувача","пароль") or die("MySQL Connection Failed");
@mysql_select_db("ім'я бази даних") or die("MySQL Database Selection Failed");
function allorders() {
$resultPurchases=mysql_query("select * from purchases");
/* отримуємо у змінну $resultPurchases всі замовлення. Ми використовуємо таку назву у тому, щоб у подальшому був плутанини, т.к. ми будемо 2 рази запитувати дані з таблиць */
$content="";
/* створюємо змінну $content, в яку будемо записувати все те, що потрібно показати користувачеві (створення HTML-файлу) */
for ($i=0; $i<mysql_num_rows($resultPurchases); $i++) {
/* перебираємо всі рядки в таблиці із замовленнями, використовуючи для цього оператор for */
$rowPurchases=mysql_fetch_array($resultPurchases);
```

```
/* отримуємо масив з даними одного ряду (одного замовлення) у змінну $rowPurchases */
$content=$content."----------<br/><br/>
<strong>".$rowPurchases['name']."</strong><br/>
".$rowPurchases['address']."<br/>
".$rowPurchases['email']."<br/>
<strong>Замовлення:</strong><br/>";
/* Складаємо інформації про покупця */
$cart=$rowPurchases['cart'];
/* записуємо в змінну cart зміст кошика */
$cartArr=explode("|",$cart);
/* використовуємо вже відому нам функцію explode, щоб отримати масив з ідентифікаторами товарів у кошику */if ($cart!="") {
/* якщо кошик не пустий */
foreach($cartArr as $k=>$v) {
$result=mysql_query("select * from products where id='$v'");
$row=mysql_fetch_array($result);
$number=$k+1;
$content=$content.$number." - ".$row['name']." - ".$row['price']." руб.<br/>";
}
/* цей фрагмент повністю повторює частину попередньої роботи – перебір усіх елементів масиву з ідентифікаторами та складання списку замовлених товарів */
}
}
return $content;
/* повертаємо змінну $content */
}
echo allorders(); /* виведемо результат роботи функції на екран */
?>
```

Якщо все зроблено правильно, то результатом роботи функції повинно бути наступне:

Призвіще Ім’я
Чернігів, Каштанова, 26
prizv@i.ua

Заказ:
1 – Чайник электричний Ч-23 – 1630 грн.
2 – М’ясорубка МК-415 – 3400 грн.

Завдання:

Написати скрипт, що дозволяє організувати електронний магазин.

Список товарів зберігається в базі даних на стороні сервера. Покупець повинен мати можливість переглянути всі наявності, що є в наявності й зробити замовлення. Покупець повинен мати можливість зробити запит, наприклад, указавши інтервал цін, який його влаштовує або які-небудь інші дані.

Доти, поки покупець вибирає окремі товари, його замовлення зберігається на стороні клієнта у вигляді cookie.

Після того як покупець сформував замовлення, замовлення відсилається на сторону сервера, де покупка товару враховується в базі даних.

Завдання для самоперевірки:

Додайте в таблицю products кілька товарів на ваш розсуд і перевірте їх наявність в каталозі.

Варіант 1

У базі даних утримується інформація про книги: автор, назва, зображення обкладинки, видавництво, рік випуску, ціна.

Варіант 2

У базі даних утримується інформація про автомобілі: модель, зображення автомобіля, рік випуску, тип кузова, потужність двигуна, колір, ціна.

Варіант 3

У базі даних утримується інформація про туристичні поїздки: країна, місто, зображення міської визначний пам'ятки, кількість днів, дата поїздки, клас готелю, ціна.

Варіант 4

У базі даних утримується інформація про журнали: назва, зображення обкладинки, рік випуску, номер, видавництво, число сторінок, ціна.

Варіант 5

У базі даних утримується інформація про місця в отеленні: назва готелю, клас номера, зображення номера, кількість місць у номері, ціна.

Зміст звіту:

- титульний аркуш;
- завдання;
- короткий опис використаних засобів і методів;
- опис бази даних;
- вихідні тексти;
- висновки по роботі.

Рекомендована література:

1 Франчук В.М. Комп’ютерні мережі та Інтернет / В.М. Франчук. // Навчально-методичний посібник для студентів фізико-математичних та інформатичних спеціальностей вищих педагогічних навчальних закладів. - К.: НПУ імені М.П. Драгоманова, 2015 р. – 141 с.

2 Мельник Р.А. Основи програмування. Конспект лекцій. — Львів: Львівська політехніка, 2009. — 80 с

3 Мельник Р.А. Програмування для мереж на основі JAVA технології. — Львів: Львівська політехніка, 2006.— 140 с.

4 Васильєв О. Програмування мовою Java.— | Тернопіль: Навчальна книга, Богдан, 2019 .— 563 с.

5 Васильєв О. Програмування мовою PHP. Навчальний посібник.— | К: Ліра-К, 2022 .— 368 с.

6 https://www.coursera.org/learn/html-css-javascript-for-web-developers

7 https://www.coursera.org/specializations/web-design