**Лабораторна робота №7**

**Тема: Встановлення HTTP-статус кодів та функції керування буферизацією виводу.**

**Мета роботи:** ознайомитися з HTTP-статусами кодів та функцієми керування буферизацією виводу.

Хід роботи:

**Завдання 1: Динамічне кешування сторінки з урахуванням статус-коду**

Створіть PHP-скрипт cache\_page.php, який реалізує кешування сторінки залежно від статус-коду відповіді.

**Умови:**

1. Якщо сторінка віддає статус 200 OK, її вміст зберігається у файлі кешу cache.html.

2. Якщо статус 404 Not Found, кешування не відбувається, а файл cache.html (якщо він існує) видаляється.

3. Якщо користувач відвідує сторінку повторно, і є кеш-файл, вміст віддається з нього без повторної генерації сторінки.

Лістинг:

Cache\_page.php

<?php

$cache\_file = 'cache.html';

if (file\_exists($cache\_file)) {

    unlink($cache\_file);

}

function generatePageContent($title, $text)

{

    $content = '<h1>'. $title . '</h1>';

    $content .= '<p>'. $text . '</p>';

    return $content;

}

// http\_response\_code(404);

if (file\_exists($cache\_file) && http\_response\_code() == 200) {

    echo file\_get\_contents($cache\_file);

} else {

    ob\_start();

    $content = generatePageContent("Some page", "Text");

    echo $content;

    $content = ob\_get\_contents();

    ob\_end\_clean();

    if (http\_response\_code() == 200) {

        file\_put\_contents($cache\_file, $content);

        echo $content;

    } elseif (http\_response\_code() == 404) {

        if (file\_exists($cache\_file)) {

            unlink($cache\_file);

        }

        echo '<h1>Сторінка не знайдена</h1>';

    }

}

.htaccess  
ErrorDocument 404 /task1/cache\_page.php

Результат виконання:



**Завдання 2. Система захисту від перевантаження сервера (Rate Limiting):**

Створіть скрипт rate\_limit.php, який обмежує кількість запитів від одного користувача та повертає відповідні статус-коди.

**Умови:**

* Логувати IP-адресу користувача в файл requests.log.
* Якщо користувач зробив більше 5 запитів за хвилину, встановлювати статус 429 Too Many Requests і рекомендувати спробувати пізніше.
* Якщо ліміт не перевищено, видавати 200 OK та звичайний контент.

Лістинг:

Rate\_limit.php

<?php

$log\_file = 'requests.log';

$limit = 5;

$time\_window = 10;

$ip = $\_SERVER['REMOTE\_ADDR'];

$current\_time = time();

$logs = file\_exists($log\_file) ? file($log\_file, FILE\_IGNORE\_NEW\_LINES | FILE\_SKIP\_EMPTY\_LINES) : [];

$filtered\_logs = [];

$request\_count = 0;

foreach ($logs as $log) {

    list($log\_ip, $log\_time) = explode('|', $log);

    if ($current\_time - $log\_time < $time\_window) {

        $filtered\_logs[] = $log;

        if ($log\_ip === $ip) {

            $request\_count++;

        }

    }

}

if ($request\_count >= $limit) {

    http\_response\_code(429);

    header('Retry-After: 10');

    die('<h1>429 Too Many Requests</h1><p>Будь ласка, спробуйте пізніше.</p>');

}

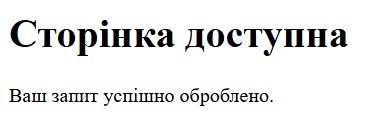
$filtered\_logs[] = "$ip|$current\_time";

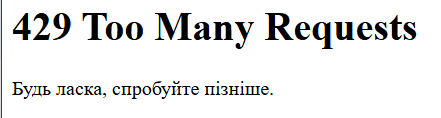
file\_put\_contents($log\_file, implode("\n", $filtered\_logs) . "\n");

http\_response\_code(200);

echo "<h1>Сторінка доступна</h1><p>Ваш запит успішно оброблено.</p>";

Результат виконання:





**Завдання 3. Перехоплення помилок та перенаправлення:**

Створіть скрипт error\_handler.php, який перехоплює фатальні помилки та замість стандартного повідомлення показує кастомну сторінку з відповідним статус-кодом.

**Умови:**

* Використовуйте register\_shutdown\_function(), щоб обробляти E\_ERROR.
* Якщо сталася помилка:
  + Очистити буфер ob\_clean()
  + Встановити статус 500 Internal Server Error
  + Показати сторінку з вибаченням та часом, коли проблема буде вирішена.
* Якщо помилок немає, віддавати 200 OK.

Лістинг:

error\_handler.php

<?php

ob\_start();

register\_shutdown\_function('shutdownHandler');

function shutdownHandler()

{

    $error = error\_get\_last();

    if ($error !== null && $error['type'] === E\_ERROR) {

        ob\_clean();

        http\_response\_code(500);

        $fix\_time = date("H:i", strtotime("+30 minutes"));

        echo "<h1>500 Internal Server Error</h1>";

        echo "<p>Вибачте, сталася помилка сервера. Очікуваний час відновлення: {$fix\_time}</p>";

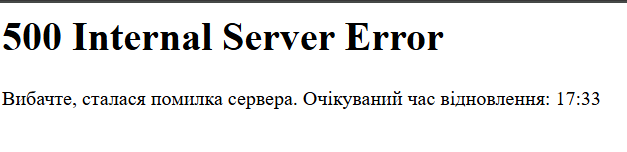
    }

}

$a = 1/0;

echo "<h1>Сторінка працює</h1><p>Все нормально!</p>";

Результат виконання:



**Завдання 4. Гнучка система редиректів:**

Створіть скрипт redirect\_manager.php, який дозволяє керувати перенаправленнями через конфігураційний файл.

**Умови:**

1. Є файл redirects.json зі списком перенаправлень:

{

"/old-page": "/new-page",

"/deprecated": "/404"

}

1. Якщо користувач заходить на /old-page, його потрібно перенаправити на /new-page зі статусом 301 Moved Permanently.
2. Якщо сторінка /deprecated, видавати 404 Not Found та відповідне повідомлення.
3. Використовувати ob\_start(), щоб контролювати вивід.

Лістинг:

Redirect\_manager.php

<?php

ob\_start();

$redirects = json\_decode(file\_get\_contents("redirects.json"), true);

$request\_url = str\_replace("/task4", "", $\_SERVER['REQUEST\_URI']);

if (isset($redirects[$request\_url])) {

    $new\_url = $redirects[$request\_url];

    if ($new\_url == "/404") {

        header("Location: /task4/404.php", true, 301);

    } else {

        header("Location: /task4" . $new\_url . ".php", true, 301);

    }

    exit;

}

http\_response\_code(200);

include("index.php");

new-page.php  
<!DOCTYPE html>

<html lang="uk">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>New Page</title>

</head>

<body>

    <h1>Це нова сторінка!</h1>

    <p>Вітаємо на новій сторінці!</p>

</body>

</html>

404.php

<!DOCTYPE html>

<html lang="uk">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>404</title>

</head>

<body>

    <h1>404 Not Found</h1>

    <p>Ця сторінка не існує.</p>

</body>

</html>

index.php

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <p><a href="/task4/old-page">Перейти на стару сторінку (old-page)</a></p>

    <p><a href="/task4/deprecated">Перейти на застарілу сторінку (deprecated)</a></p>

</body>

</html>

.htaccess

RewriteEngine On

RewriteCond %{REQUEST\_FILENAME} !-f

RewriteCond %{REQUEST\_FILENAME} !-d

RewriteRule ^(.\*)$ redirect\_manager.php [L,QSA]

Redirects.json

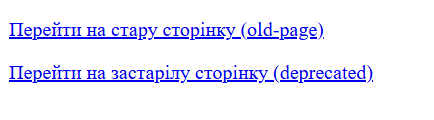
{

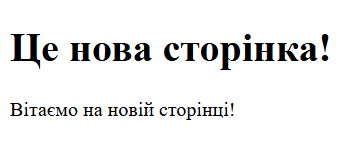
    "/old-page": "/new-page",

    "/deprecated": "/404"

}

Результат виконання:







**Завдання 5. Динамічне підключення заголовків без видимого виводу:**

Створіть PHP-клас Response, який допомагає керувати HTTP-заголовками та буферизацією.

**Функціонал класу:**

* Метод setStatus($code), який встановлює HTTP-статус.
* Метод addHeader($header), який додає заголовок (наприклад, Content-Type).
* Метод send($content), який очищає буфер, додає заголовки та відправляє відповідь.

Лістинг:

Response.php

<?php

class Response

{

    private int $statusCode;

    private array $headers = [];

    public function setStatus(int $code): void

    {

        $this->statusCode = $code;

        header("HTTP/1.1 $code");

    }

    public function addHeader(string $header): void

    {

        $this->headers[] = $header;

    }

    public function send(string $content): void

    {

        ob\_start();

        ob\_clean();

        foreach ($this->headers as $header) {

            header($header);

        }

        echo $content;

        ob\_end\_flush();

    }

}

Index.php

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <?php

    require\_once 'Response.php';

    $response = new Response();

    $response->setStatus(200);

    $response->addHeader("Content-Type: text/html");

    $response->send("<h1>Вітаємо!</h1><p>Це динамічна відповідь.</p>");

    $response = new Response();

    $response->setStatus(404);

    $response->addHeader("Content-Type: text/html");

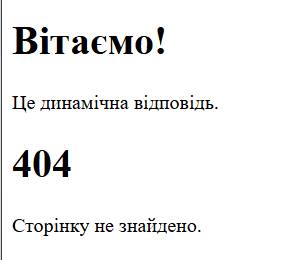
    $response->send("<h1>404</h1><p>Сторінку не знайдено.</p>");

    ?>

</body>

</html>

Результат виконання:



**Завдання 6. Аналіз трафіку та відповідей сервера:**

Розробіть traffic\_logger.php, який записує інформацію про всі запити в базу даних.

**Що зберігати в БД?**

* IP-адресу
* Час запиту
* Запитаний URL
* Відповідний HTTP-статус

**Додаткові функції:**

1. Реалізувати перегляд статистики через stats.php, який рахує кількість 404 помилок за останню добу.
2. Якщо відсоток 404 помилок перевищує 10% від загальної кількості запитів, надсилати адміністратору повідомлення.

Лістинг:

Index.php

<?php include 'traffic\_logger.php'; ?>

<!DOCTYPE html>

<html lang="uk">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Головна сторінка</title>

</head>

<body>

    <h1>Ласкаво просимо!</h1>

    <p>Це головна сторінка нашого сайту.</p>

    <div>

        <a href="about.php">Про нас</a>

        <a href="stats.php">Статистика</a>

    </div>

</body>

</html>

About.php

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <h1>Про нас</h1>

    <a href='index.php'>На головну</a>

</body>

</html>

404.php

<?php

http\_response\_code(404);

include 'traffic\_logger.php';

?>

<!DOCTYPE html>

<html lang="uk">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Сторінка не знайдена</title>

</head>

<body>

    <div class="container">

        <h1>404 - Сторінка не знайдена</h1>

        <p>Вибачте, але запитана сторінка не існує.</p>

        <a href="index.php">На головну</a>

    </div>

</body>

</html>

Stats.php

<?php

$db = new PDO('mysql:host=localhost;dbname=lab7;charset=utf8', 'root', '');

$db->setAttribute(PDO::ATTR\_ERRMODE, PDO::ERRMODE\_EXCEPTION);

$totalStmt = $db->prepare("SELECT COUNT(\*) FROM logs WHERE request\_time >= NOW() - INTERVAL 1 DAY");

$totalStmt->execute();

$totalRequests = $totalStmt->fetchColumn();

$errorStmt = $db->prepare("SELECT COUNT(\*) FROM logs WHERE status = 404 AND request\_time >= NOW() - INTERVAL 1 DAY");

$errorStmt->execute();

$errorRequests = $errorStmt->fetchColumn();

$percent404 = ($totalRequests > 0) ? ($errorRequests / $totalRequests) \* 100 : 0;

echo "<h2>Статистика 404 помилок за останню добу</h2>";

echo "<p>Загальна кількість запитів: $totalRequests</p>";

echo "<p>Кількість 404 помилок: $errorRequests</p>";

echo "<p>Відсоток 404 помилок: " . number\_format($percent404, 2) . "%</p>";

echo "<a href='index.php'>На головну</a>";

if ($percent404 > 10) {

    echo "<h3>Попередження: Відсоток 404 помилок перевищив 10%! Перевірте сервер.</р>";

}

Traffic\_logger.php

<?php

$db = new PDO('mysql:host=localhost;dbname=lab7;charset=utf8', 'root', '');

$db->setAttribute(PDO::ATTR\_ERRMODE, PDO::ERRMODE\_EXCEPTION);

$ip = $\_SERVER['REMOTE\_ADDR'];

$url = $\_SERVER['REQUEST\_URI'];

$time = date('Y-m-d H:i:s');

$status = http\_response\_code();

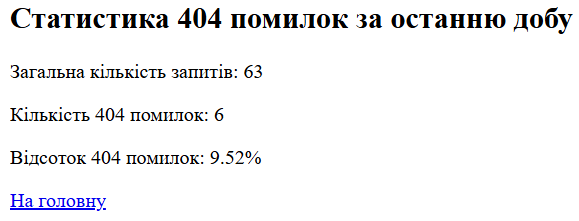
$stmt = $db->prepare("INSERT INTO logs (ip, request\_time, url, status) VALUES (?, ?, ?, ?)");

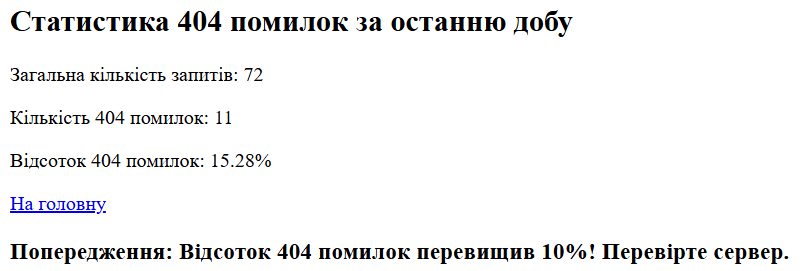
$stmt->execute([$ip, $time, $url, $status]);

.htaccess

ErrorDocument 404 /task6/404.php

Результат виконання:





Висновок: Під час виконання лабораторної роботи ознайомилися з HTTP-статусами кодів та функцієми керування буферизацією виводу.