**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:

<?php

ob\_start();

header("Content-Type: text/html; charset=UTF-8");

$status = 200;

$content = "<h1>Це контент сторінки</h1>";

$cache\_file = "cache.html";

if (isset($\_GET['status']) && $\_GET['status'] == 404) {

    $status = 404;

    $content = "<h1>Сторінку не знайдено</h1>";

    if (file\_exists($cache\_file)) unlink($cache\_file);

    http\_response\_code(404);

} elseif (file\_exists($cache\_file)) {

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

    exit;

} else {

    http\_response\_code(200);

    echo $content;

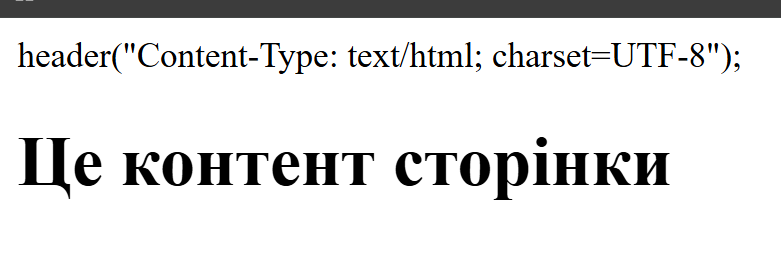
    file\_put\_contents($cache\_file, ob\_get\_contents());

}

ob\_end\_flush();

?>

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:

<?php

ob\_start();

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

$log\_file = "requests.log";

$limit = 5;

$interval = 60;

$time = time();

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

$entries = array\_filter($entries, function($line) use ($time, $interval) {

    $parts = explode("|", $line);

    if (count($parts) < 2) return false;

    return ($time - (int)$parts[1]) < $interval;

});

file\_put\_contents($log\_file, implode("\n", $entries) . "\n", LOCK\_EX);

$entry\_count = 0;

foreach ($entries as $entry) {

    $parts = explode("|", $entry);

    if (count($parts) >= 2 && $parts[0] === $ip) $entry\_count++;

}

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

    http\_response\_code(429);

    echo "Забагато запитів. Спробуйте пізніше.";

} else {

    file\_put\_contents($log\_file, "$ip|$time\n", FILE\_APPEND);

    http\_response\_code(200);

    echo "Ви можете переглянути сторінку.";

}

ob\_end\_flush();

?>

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();

function shutdownHandler() {

    $error = error\_get\_last();

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

        ob\_clean();

        http\_response\_code(500);

        echo "<h1>500 Internal Server Error</h1><p>Ми працюємо над вирішенням проблеми.</p>";

    }

}

register\_shutdown\_function('shutdownHandler');

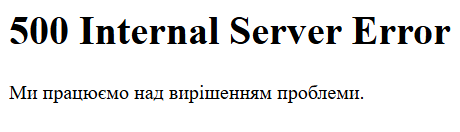
// Фатальна помилка для тесту

undefined\_function();

ob\_end\_flush();

?>

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:

<?php

ob\_start();

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

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

if (isset($redirects[$uri])) {

    $target = $redirects[$uri];

    if ($target === "/404") {

        http\_response\_code(404);

        echo "Сторінку не знайдено.";

    } else {

        header("Location: $target", true, 301);

        exit;

    }

} else {

    echo "Поточна сторінка: $uri";

}

ob\_end\_flush();

?>

{

  "/old-page": "/new-page",

  "/deprecated": "/404",

  "/start": "/task1/cache\_page.php",

  "/limit": "/task2/rate\_limit.php"

}

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:

<?php

require\_once "Response.php";

ob\_start();

$response = new Response();

$response->setStatus(200);

    $response->addHeader("Content-Type: text/html; charset=UTF-8");

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

?>

<?php

class Response {

    private $status;

    private $headers = [];

    public function setStatus($code) {

        $this->status = $code;

        http\_response\_code($code);

    }

    public function addHeader($header) {

        $this->headers[] = $header;

    }

    public function send($content) {

        ob\_clean();

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

            header($header);

        }

        echo $content;

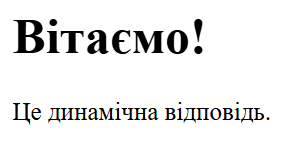
        ob\_end\_flush();

    }

}

?>

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:

<?php

$conn = new mysqli("localhost", "root", "", "lab7");

if ($conn->connect\_error) {

    die("Помилка з'єднання з базою даних");

}

$res = $conn->query("SELECT COUNT(\*) as total FROM traffic\_logs WHERE created\_at >= NOW() - INTERVAL 1 DAY");

$total = $res->fetch\_assoc()['total'] ?? 0;

$res404 = $conn->query("SELECT COUNT(\*) as err FROM traffic\_logs WHERE status\_code=404 AND created\_at >= NOW() - INTERVAL 1 DAY");

$errors = $res404->fetch\_assoc()['err'] ?? 0;

echo "404 за добу: $errors з $total запитів<br>";

if ($total > 0 && ($errors / $total) > 0.1) {

    echo "⚠️ Понад 10% 404 помилок! Повідомлено адміністратора.";

}

?>

<?php

$conn = new mysqli("localhost", "root", "", "lab7");

if ($conn->connect\_error) {

    die("Помилка з'єднання");

}

$stmt = $conn->prepare("INSERT INTO traffic\_logs (url, status\_code, created\_at) VALUES (?, ?, NOW())");

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

$code = http\_response\_code(); // або вручну 404

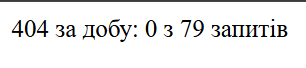
$stmt->bind\_param("si", $url, $code);

$stmt->execute();

echo "Записано.";

?>

Результат виконання:





**Висновок**

Під час виконання лабораторної роботи №7 я детально ознайомився з поняттям HTTP-статус кодів, зокрема 200 (OK), 404 (Not Found), 500 (Server Error) та іншими, і навчився встановлювати їх у PHP за допомогою функції http\_response\_code(). Було розглянуто буферизацію виводу через ob\_start() і ob\_end\_flush(), що дозволяє контролювати, коли саме дані виводяться на клієнт. На практиці я реалізував кешування HTML-вмісту, обробку помилок, а також створив простий логгер запитів і систему статистики. Отримані знання є корисними для побудови надійних і ефективних веб-додатків.