

Дисципліна

«Бекенд-розробка, сервіси та мікросервіси»

КОНСПЕКТ ЛЕКЦІЇ. ТЕМА 5

Вихідний буфер, формати обміну даними JSON та XML, робота з файлами, виконання HTTP-запитів через cURL

Спеціальність: Комп'ютерна інженерія, 3 курс
Обсяг: 1 год 20 хв (лекція 5 з 18)

План лекції

1. Вихідний буфер (output buffering)
2. Формат обміну даними JSON
3. Формат обміну даними XML
4. Робота з файлами: читання та запис
5. Робота з файлами: практичні операції
6. Виконання HTTP-запитів через cURL
7. cURL на практиці: робота з JSON API

Мета лекції: навчитися керувати виводом PHP-скрипта, обмінюватися структурованими даними у форматах JSON і XML, читати й записувати файли на сервері та отримувати дані із зовнішніх джерел через HTTP-запити засобами cURL — усе це базові навички для інтеграції бекенду із зовнішнім світом.

1. Вихідний буфер (output buffering)

За замовчуванням PHP надсилає вивід (усе, що виводить echo чи print) браузеру одразу, рядок за рядком, у міру виконання скрипта. Вихідний буфер (output buffer) дозволяє тимчасово «накопичувати» цей вивід у пам'яті сервера замість негайного надсилання — це відкриває кілька практичних можливостей, які без буферизації неможливі.

1.1. Основні функції

Функція	Призначення
ob_start()	Починає буферизацію виводу
ob_get_clean()	Повертає вміст буфера як рядок і очищує (закриває) буфер
ob_end_flush()	Надсилає вміст буфера клієнту і закриває буфер
ob_end_clean()	Видаляє вміст буфера без надсилання клієнту
ob_get_contents()	Повертає вміст буфера, не закриваючи його

```
<?php
ob_start();

echo "<h1>Привіт!</h1>";
echo "<p>Це вміст сторінки.</p>";

$html = ob_get_clean(); // забираємо вивід у змінну, нічого не надіслано браузеру

echo strtoupper($html); // можемо обробити перед фактичним виводом
```

1.2. Навіщо потрібна буферизація

- Зміна заголовків після початку виводу. Функція header() працює, лише поки жоден байт вмісту ще не надіслано; ob_start() дозволяє «накопичити» HTML і все ж викликати header() пізніше (наприклад, для редиректу після часткового виводу).
- Кешування сторінок. Згенерований HTML можна зберегти у файл через ob_get_contents() і за наступних запитів віддавати кешовану версію, не виконуючи важку логіку повторно.
- Обробка виводу сторонніх бібліотек чи include-файлів, які самі викликають echo, коли потрібно отримати їхній результат як рядок (наприклад, для вставки в шаблон чи PDF-документ).

Типова помилка "headers already sent" виникає саме тому, що PHP уже почав надсилати вивід (навіть один пробіл чи порожній рядок перед <?php у файлі) до виклику header() чи setcookie(). Обгортання коду в ob_start() на початку скрипта — поширений спосіб уникнути цієї помилки.

2. Формат обміну даними JSON

JSON (JavaScript Object Notation) — легкий текстовий формат обміну структурованими даними, який де-факто став стандартом для REST API: більшість сучасних вебсервісів приймають і повертають дані саме в цьому форматі. PHP має вбудовану підтримку JSON без будь-яких додаткових бібліотек.

2.1. Кодування: `json_encode()`

Функція `json_encode()` перетворює масив або об'єкт PHP на рядок у форматі JSON.

```
<?php
$user = [
    "name" => "Оксана",
    "age" => 25,
    "skills" => ["PHP", "MySQL", "JavaScript"],
];

echo json_encode($user);
// {"name":"Оксана","age":25,"skills":["PHP","MySQL","JavaScript"]}

// Прапорці форматування
echo json_encode($user, JSON_PRETTY_PRINT | JSON_UNESCAPED_UNICODE);
```

Прапорець	Ефект
JSON_PRETTY_PRINT	Форматує результат з відступами для читабельності
JSON_UNESCAPED_UNICODE	Не екранує кирилицю та інші не-ASCII символи як \uXXXX
JSON_UNESCAPED_SLASHES	Не екранує символ / як \

2.2. Декодування: `json_decode()`

Функція `json_decode()` виконує зворотну операцію — перетворює рядок JSON на дані PHP. За замовчуванням вона повертає об'єкт `stdClass`; другий аргумент `true` змушує повернути асоціативний масив, з яким зазвичай зручніше працювати.

```
<?php
$json = '{"name":"Іван","age":30}';

$obj = json_decode($json);
echo $obj->name;           // Іван (об'єкт stdClass)

$arr = json_decode($json, true);
echo $arr['name'];         // Іван (асоціативний масив)
```

Якщо переданий рядок не є коректним JSON, `json_decode()` поверне `null`, а `json_encode()` поверне `false` у разі помилки (наприклад, якщо масив містить некоректну кодову послідовність). Функція `json_last_error()` повертає код останньої помилки — перевірку варто робити завжди, коли дані надходять із зовнішнього джерела.

3. Формат обміну даними XML

XML (eXtensible Markup Language) — старіший за JSON формат структурованих даних на основі вкладених тегів. Хоча в нових API JSON майже витіснив XML через компактність і простоту, XML досі широко використовується в корпоративних системах, SOAP-сервісах, RSS/Atom-стрічках і конфігураційних файлах, тому вміння з ним працювати залишається важливим.

3.1. Читання XML через SimpleXML

Розширення SimpleXML перетворює XML-документ на об'єкт, властивості якого відповідають вкладеним тегам — це найпростіший спосіб прочитати нескладний XML-файл.

```
<?php
$xmlString = <<<XML
<product>
    <name>Ноутбук</name>
    <price>25000</price>
</product>
XML;

$product = simplexml_load_string($xmlString);
echo $product->name;    // Ноутбук
echo $product->price;    // 25000
```

3.2. Створення й редагування через DOMDocument

Коли потрібно не лише читати, а й програмно створювати чи модифікувати XML, використовують клас DOMDocument — він надає повний контроль над структурою документа (створення елементів, атрибутів, вкладеність).

```
<?php
$dom = new DOMDocument('1.0', 'UTF-8');
$dom->formatOutput = true;

$product = $dom->createElement('product');
$name = $dom->createElement('name', 'Ноутбук');
$price = $dom->createElement('price', '25000');

$product->appendChild($name);
$product->appendChild($price);
$dom->appendChild($product);

echo $dom->saveXML();
```

Для вибірки елементів за складнішими критеріями (наприклад, «усі товари дорожчі за 1000») зручно використовувати XPath — мову запитів до XML-документа, доступну через клас `DOMXPath`. Це виходить за межі сьогоднішньої лекції, але варто знати цей інструмент для подальшого самостійного вивчення.

4. Робота з файлами: читання та запис

4.1. Швидкі операції: `file_get_contents` і `file_put_contents`

Для простих сценаріїв (прочитати весь файл або записати рядок цілком) PHP надає дві зручні функції, які самі відкривають і закривають файл.

```
<?php
// Читання всього файлу в рядок
$content = file_get_contents('data.txt');
echo $content;

// Запис рядка у файл (перезаписує вміст)
file_put_contents('data.txt', "Новий вміст файлу");

// Додавання в кінець файлу, не перезаписуючи
file_put_contents('log.txt', "Новий запис\n", FILE_APPEND);
```

4.2. Покроковий доступ: `fopen`, `fread`, `fwrite`, `fclose`

Коли файл великий або потрібен більш тонкий контроль (читання по рядку, запис у процесі обробки), використовують низькорівневі функції: `fopen()` відкриває файл і повертає дескриптор (resource), з яким далі працюють `fread()/fwrite()/fgets()`, а `fclose()` обов'язково закриває файл після завершення роботи.

Режим <code>fopen()</code>	Значення
'r'	Лише читання, вказівник на початок файлу
'w'	Лише запис; існуючий вміст видаляється
'a'	Лише запис; вказівник у кінці файлу (дописування)
'r+'	Читання і запис, вказівник на початку

```
<?php
$handle = fopen('data.txt', 'r');

while (!feof($handle)) {
    $line = fgets($handle);    // читає один рядок
    echo $line;
}

fclose($handle);

// Запис по частинах
$out = fopen('output.txt', 'w');
fwrite($out, "Перший рядок\n");
fwrite($out, "Другий рядок\n");
fclose($out);
```

fclose() легко забути, особливо коли в коді трапляється помилка до того, як виконання доходить до цього рядка — файл лишається відкритим ("висить"), займаючи ресурс. Для критичного коду варто оточувати роботу з файлом конструкцією try/finally, щоб fclose() виконався в будь-якому разі.

5. Робота з файлами: практичні операції

5.1. Перевірка та керування файлами

Функція	Призначення
file_exists(\$path)	Перевіряє, чи існує файл або тека
is_file(\$path) / is_dir(\$path)	Перевіряє, чи є шлях файлом / текою
unlink(\$path)	Видаляє файл
mkdir(\$path)	Створює нову теку
copy(\$from, \$to)	Копіює файл
rename(\$from, \$to)	Перейменовує або переміщує файл
filesize(\$path)	Повертає розмір файлу в байтах

5.2. Робота з CSV-файлами

CSV (Comma-Separated Values) — поширений текстовий формат для табличних даних (наприклад, вивантаження з Excel). PHP має вбудовані функції для читання й запису CSV без ручного розбору рядків через explode().

```
<?php
// Читання CSV по рядку
$handle = fopen('users.csv', 'r');
while (($row = fgetcsv($handle)) !== false) {
    // $row — масив значень одного рядка
    echo $row[0] . ' — ' . $row[1] . "\n";
}
fclose($handle);

// Запис у CSV
$out = fopen('export.csv', 'w');
fputcsv($out, ['Ім'я', 'Вік', 'Місто']);
fputcsv($out, ['Оксана', 25, 'Львів']);
fclose($out);
```

5.3. Збереження структурованих даних у JSON-файл

Поєднання роботи з файлами й JSON — типовий сценарій для простого зберігання даних без бази даних (наприклад, налаштувань застосунку чи невеликого кешу).

```
<?php
$settings = ['theme' => 'dark', 'language' => 'uk'];
file_put_contents('settings.json', json_encode($settings,
JSON_PRETTY_PRINT));
```

```
// Пізніше, при наступному запуску:  
$settings = json_decode(file_get_contents('settings.json'), true);  
echo $settings['theme'];    // dark
```

6. Виконання HTTP-запитів через cURL

cURL — бібліотека для виконання мережевих запитів за різними протоколами, найчастіше HTTP/HTTPS. У PHP розширення cURL дозволяє серверному скрипту самому виступати клієнтом і звертатися до зовнішніх API — отримувати курс валют, надсилати повідомлення через сервіс розсилок, інтегруватися зі сторонніми сервісами тощо.

6.1. Базовий GET-запит

Робота з cURL завжди складається з однакової послідовності кроків: ініціалізація сесії, встановлення параметрів, виконання запиту, закриття сесії.

```
<?php
$ch = curl_init();

curl_setopt($ch, CURLOPT_URL, 'https://api.example.com/users/1');
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true); // повернути результат,
а не вивести одразу

$response = curl_exec($ch);

if (curl_errno($ch)) {
    echo 'Помилка cURL: ' . curl_error($ch);
} else {
    $data = json_decode($response, true);
    echo $data['name'];
}

curl_close($ch);
```

Опція CURLOPT_*	Призначення
URL	Адреса запиту
RETURNTRANSFER	true — повернути відповідь як рядок замість виводу на екран
POST	true — виконати запит методом POST
POSTFIELDS	Дані, що надсилаються в тілі запиту
HTTPHEADER	Масив користувацьких заголовків запиту
TIMEOUT	Максимальний час очікування відповіді (секунди)

7. cURL на практиці: робота з JSON API

Сучасні API майже завжди очікують і повертають JSON. Типовий запит до такого API — POST з JSON-тілом і відповідним заголовком Content-Type.

```
<?php
$payload = json_encode([
    'title' => 'Нова задача',
    'completed' => false,
]);

$ch = curl_init('https://api.example.com/tasks');

curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
curl_setopt($ch, CURLOPT_POST, true);
curl_setopt($ch, CURLOPT_POSTFIELDS, $payload);
curl_setopt($ch, CURLOPT_HTTPHEADER, [
    'Content-Type: application/json',
    'Authorization: Bearer ' . $apiToken,
]);

$response = curl_exec($ch);
$statusCode = curl_getinfo($ch, CURLINFO_HTTP_CODE);
curl_close($ch);

if ($statusCode === 201) {
    $created = json_decode($response, true);
    echo "Створено задачу #" . $created['id'];
}
```

curl_getinfo(\$ch, CURLINFO_HTTP_CODE) повертає HTTP-статус відповіді (200, 201, 404, 500 тощо) — перевірка статус-коду настільки ж важлива, як перевірка curl_errno(): запит може успішно "доїхати" до сервера (без мережевої помилки), але сервер поверне 404 чи 401, і це потрібно обробити окремо від помилок з'єднання.

Питання для самоперевірки

- Для чого потрібна буферизація виводу і чому без неї виклик `header()` після `echo` призводить до помилки?
- Чим `json_decode($json, true)` відрізняється від `json_decode($json)` без другого аргументу?
- Наведіть по одному сценарію, де XML досі використовується частіше за JSON.
- Чим відрізняються режими `foren()` 'w' і 'a'? Що станеться з існуючим вмістом файлу в кожному випадку?
- Для чого призначені `fgetcsv()` і `fputcsv()`? Чим вони зручніші за ручний розбір рядків через `explode(",")`?
- Опишіть послідовність кроків типового cURL-запиту від `curl_init()` до `curl_close()`.
- Чим `curl_errno()` відрізняється від перевірки HTTP-статусу через `curl_getinfo(..., CURLINFO_HTTP_CODE)`?
- Як передати JSON-тіло в POST-запиті cURL і який заголовок для цього потрібно встановити?

Короткий глосарій

Термін	Означення
Вихідний буфер (output buffer)	Тимчасове сховище виводу скрипта в пам'яті сервера перед надсиланням клієнту
JSON	Легкий текстовий формат обміну структурованими даними на основі пар ключ-значення
XML	Текстовий формат структурованих даних на основі вкладених тегів
Дескриптор файлу (resource)	Посилання на відкритий файл, яке повертає fopen()
CSV	Текстовий формат табличних даних, розділених комами
cURL	Бібліотека для виконання мережових запитів за різними протоколами
HTTP-статус	Тризначний код відповіді сервера (200, 404, 500 тощо)