Дисципліна

«Бекенд-розробка, сервіси та мікросервіси»

КОНСПЕКТ ЛЕКЦІЇ. ТЕМА 6

Опрацювання HTTP-запитів, отримання даних із форм (параметри та JSON), завантаження файлів на сервер

Спеціальність: Комп'ютерна інженерія, 3 курс
Обсяг: 1 год 20 хв (лекція 6 з 18)

План лекції

1. Опрацювання HTTP-запитів: методи та суперглобальні масиви
2. Отримання даних форми у вигляді параметрів
3. Отримання даних у форматі JSON
4. Валідація та фільтрація вхідних даних
5. Завантаження файлів на сервер: механізм і $_FILES
6. Завантаження файлів: перевірка та збереження
7. Завантаження кількох файлів одночасно

> *Мета лекції: навчитися приймати й коректно обробляти дані, які надсилає клієнт — як прості параметри форми, так і структуровані JSON-дані, а також організувати безпечне завантаження файлів на сервер.*

1. Опрацювання HTTP-запитів: методи та суперглобальні масиви

Будь-який PHP-скрипт, що обробляє веб-запит, починає роботу з визначення того, який метод HTTP використав клієнт і які дані він передав. Метод запиту завжди доступний через суперглобальний масив $_SERVER.

```
<?php
$method = $_SERVER['REQUEST_METHOD'];   // 'GET', 'POST', 'PUT',
'DELETE' ...

if ($method === 'POST') {
    // обробка даних форми
} elseif ($method === 'GET') {
    // обробка параметрів рядка запиту
}
```

Нагадаємо суперглобальні масиви, відповідальні за вхідні дані запиту (детально розглянуті в темі 2): $_GET містить параметри рядка запиту (?key=value), $_POST — дані, надіслані у тілі запиту з Content-Type application/x-www-form-urlencoded або multipart/form-data, а $_REQUEST об'єднує обидва (та ще й $_COOKIE), хоча на практиці через неоднозначність джерела його використовують рідко.

> *Важливо розрізняти: $_POST заповнюється автоматично лише для двох конкретних Content-Type (form-urlencoded і multipart/form-data). Якщо клієнт надсилає POST-запит із тілом у форматі JSON (Content-Type: application/json), масив $_POST залишиться порожнім — дані потрібно читати інакше (розділ 3).*

2. Отримання даних форми у вигляді параметрів

Класична HTML-форма без явно вказаного enctype надсилає дані у форматі application/x-www-form-urlencoded — кожне поле форми стає парою ключ-значення, доступною через $_POST (для методу POST) або $_GET (для методу GET).

```
<!-- HTML-форма -->
<form method="POST" action="submit.php">
    <input type="text" name="username">
    <input type="email" name="email">
    <button type="submit">Надіслати</button>
</form>
```

```
<?php
// submit.php
$username = $_POST['username'] ?? '';
$email = $_POST['email'] ?? '';

echo "Отримано: $username, $email";
```

Оператор ?? (null coalescing) тут відіграє важливу роль: якщо поле не було надіслано (наприклад, форму відправили без заповнення), звернення до неіснуючого ключа масиву без ?? згенерувало б попередження (Warning: Undefined array key).

2.1. Масиви полів форми

Якщо ім'я поля форми закінчується на [] (наприклад, name="tags[]"), PHP автоматично збирає всі однойменні поля в масив — це зручно для чекбоксів або множинного вибору.

```
<!-- Кілька чекбоксів з однаковим базовим іменем -->
<input type="checkbox" name="tags[]" value="php">
<input type="checkbox" name="tags[]" value="mysql">
```

```
<?php
$tags = $_POST['tags'] ?? [];   // масив, напр. ['php', 'mysql']
foreach ($tags as $tag) {
    echo $tag . "\n";
}
```

3. Отримання даних у форматі JSON

Сучасні клієнти (JavaScript-застосунки через fetch/axios, мобільні застосунки, інші сервіси) часто надсилають дані у форматі JSON із заголовком Content-Type: application/json. Оскільки це не один із двох форматів, які PHP розбирає автоматично, дані потрібно прочитати вручну з так званого «сирого» тіла запиту.

3.1. php://input

php://input — спеціальний потік, який дає доступ до необробленого (сирого) тіла HTTP-запиту, незалежно від його Content-Type. Це єдиний надійний спосіб прочитати JSON, XML чи будь-який інший формат тіла запиту, який PHP не розбирає автоматично.

```
<?php
// Клієнт надіслав: {"name":"Оксана","age":25}
// з заголовком Content-Type: application/json

$rawBody = file_get_contents('php://input');
$data = json_decode($rawBody, true);

if (json_last_error() !== JSON_ERROR_NONE) {
    http_response_code(400);
    echo json_encode(['error' => 'Некоректний JSON']);
    exit;
}

$name = $data['name'] ?? null;
$age = $data['age'] ?? null;
```

> *php://input можна прочитати лише один раз за запит (це потік, а не змінна) — якщо потрібно використати сирі дані в кількох місцях коду, збережіть результат file_get_contents() у змінну одразу й далі працюйте з нею.*

3.2. Формування JSON-відповіді

Коли бекенд працює як API, відповідь клієнту також зазвичай формується у форматі JSON, разом із коректним заголовком Content-Type і HTTP-статусом.

```
<?php
header('Content-Type: application/json');
http_response_code(201);   // Created

echo json_encode([
    'status' => 'success',
    'user' => ['id' => 42, 'name' => $name],
]);
```

4. Валідація та фільтрація вхідних даних

Дані, що надходять від клієнта — з форми, JSON-тіла чи параметрів URL — ніколи не можна вважати безпечними чи коректними за замовчуванням: користувач (або зловмисник) може надіслати будь-що. Перш ніж використовувати такі дані, їх слід перевірити (валідація) і, за потреби, очистити (санітизація/фільтрація).

4.1. Розширення filter

Вбудована функція filter_var() перевіряє чи очищує значення відповідно до заданого фільтра — це надійніший спосіб, ніж писати власні регулярні вирази для типових перевірок (email, URL, числа).

Фільтр	Призначення
FILTER_VALIDATE_EMAIL	Перевіряє коректність формату e-mail
FILTER_VALIDATE_INT	Перевіряє й перетворює на ціле число
FILTER_VALIDATE_FLOAT	Перевіряє й перетворює на дійсне число
FILTER_VALIDATE_URL	Перевіряє коректність формату URL
FILTER_SANITIZE_STRING / htmlspecialchars	Видаляє чи екранує потенційно небезпечні теги/символи

```
<?php
$email = $_POST['email'] ?? '';

if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
    http_response_code(422);
    echo json_encode(['error' => 'Некоректний email']);
    exit;
}

// Безпечний вивід тексту, введеного користувачем, у HTML
$comment = $_POST['comment'] ?? '';
echo htmlspecialchars($comment, ENT_QUOTES, 'UTF-8');
```

> *htmlspecialchars() перетворює спецсимволи HTML (<, >, ", ') на їхні безпечні екрановані відповідники — це основний захист від XSS (Cross-Site Scripting), коли введений користувачем текст виводиться назад на сторінку. Правило: ніколи не виводьте вхідні дані користувача в HTML без екранування.*

5. Завантаження файлів на сервер: механізм і $_FILES

Щоб форма могла надсилати файли, вона повинна мати атрибут enctype="multipart/form-data" — цей формат кодування дозволяє передавати бінарні дані (вміст файлу) разом зі звичайними текстовими полями в одному запиті.

```
<!-- HTML-форма для завантаження файлу -->
<form method="POST" action="upload.php" enctype="multipart/form-data">
    <input type="file" name="avatar">
    <button type="submit">Завантажити</button>
</form>
```

Інформація про завантажений файл потрапляє в суперглобальний масив $_FILES, ключем якого є ім'я поля форми (у прикладі — "avatar"). Значенням є вкладений асоціативний масив із метаданими файлу.

Ключ $_FILES['avatar'][...]	Зміст
['name']	Оригінальне ім'я файлу на комп'ютері користувача
['type']	MIME-тип, який повідомив браузер (не варто довіряти беззастережно)
['tmp_name']	Шлях до тимчасового файлу на сервері, куди PHP вже зберіг вміст
['error']	Код помилки завантаження (0 — успіх, розділ 6)
['size']	Розмір файлу в байтах

```
<?php
// upload.php
print_r($_FILES['avatar']);
// Array (
//     [name] => photo.jpg
//     [type] => image/jpeg
//     [tmp_name] => /tmp/phpA1b2C3
//     [error] => 0
//     [size] => 204800
// )
```

> *PHP автоматично зберігає завантажений файл у тимчасову теку сервера ще до того, як ваш код почне виконуватись — цей тимчасовий файл видаляється наприкінці запиту, якщо його явно не перемістити функцією move_uploaded_file() (розділ 6).*

6. Завантаження файлів: перевірка та збереження

6.1. Коди помилок завантаження

Перш ніж працювати з файлом, обов'язково перевіряють значення ['error'] — воно показує, чи завантаження взагалі пройшло успішно.

Константа	Значення	Причина
UPLOAD_ERR_OK	0	Файл завантажено успішно
UPLOAD_ERR_INI_SIZE	1	Файл перевищує upload_max_filesize у php.ini
UPLOAD_ERR_FORM_SIZE	2	Файл перевищує MAX_FILE_SIZE, вказаний у формі
UPLOAD_ERR_PARTIAL	3	Файл завантажено лише частково
UPLOAD_ERR_NO_FILE	4	Файл не було обрано

6.2. Переміщення файлу: move_uploaded_file()

Тимчасовий файл потрібно перемістити у постійне місце на сервері функцією move_uploaded_file() — вона, на відміну від звичайного rename(), додатково перевіряє, що файл дійсно було завантажено через HTTP POST (захист від підміни шляху).

```php
<?php
$file = $_FILES['avatar'];

if ($file['error'] !== UPLOAD_ERR_OK) {
    http_response_code(400);
    exit('Помилка завантаження файлу');
}

// Перевірка розміру (наприклад, не більше 2 МБ)
if ($file['size'] > 2 * 1024 * 1024) {
    exit('Файл завеликий (максимум 2 МБ)');
}

// Перевірка розширення й реального MIME-типу файлу
$allowedTypes = ['image/jpeg', 'image/png'];
$actualType = mime_content_type($file['tmp_name']);
if (!in_array($actualType, $allowedTypes, true)) {
    exit('Дозволені лише JPEG та PNG');
}

// Генеруємо унікальне ім'я, щоб уникнути конфліктів і небезпечних імен
$extension = pathinfo($file['name'], PATHINFO_EXTENSION);
$newName = uniqid('avatar_', true) . '.' . $extension;
$destination = __DIR__ . '/uploads/' . $newName;

if (move_uploaded_file($file['tmp_name'], $destination)) {
    echo "Файл збережено як $newName";
} else {
    echo "Не вдалося зберегти файл";
}
```

> *Перевірка MIME-типу через ['type'] із $_FILES ненадійна — це значення повідомляє браузер клієнта, і його легко підробити. Функція mime_content_type() (або розширення fileinfo) перевіряє реальний вміст файлу на сервері, що значно безпечніше. Також ніколи не використовуйте оригінальне ім'я файлу від користувача без обробки — воно може містити символи для обходу шляху (../../etc/passwd).*

7. Завантаження кількох файлів одночасно

Якщо поле файлу в формі має атрибут multiple та ім'я із [] (наприклад, name="photos[]"), $_FILES набуває дещо іншої, «вивернутої» структури: замість масиву файлів PHP створює масив, де кожен ключ (name, type, tmp_name, error, size) сам містить масив значень для кожного файлу.

```
<!-- Форма з можливістю обрати кілька файлів -->
<form method="POST" action="upload.php" enctype="multipart/form-data">
    <input type="file" name="photos[]" multiple>
    <button type="submit">Завантажити</button>
</form>
```

```
<?php
// $_FILES['photos']['name']     — масив імен усіх файлів
// $_FILES['photos']['tmp_name'] — масив тимчасових шляхів

$count = count($_FILES['photos']['name']);

for ($i = 0; $i < $count; $i++) {
    if ($_FILES['photos']['error'][$i] !== UPLOAD_ERR_OK) {
        continue;   // пропускаємо файл із помилкою
    }

    $tmpName = $_FILES['photos']['tmp_name'][$i];
    $originalName = $_FILES['photos']['name'][$i];
    $extension = pathinfo($originalName, PATHINFO_EXTENSION);
    $newName = uniqid('photo_', true) . '.' . $extension;

    move_uploaded_file($tmpName, __DIR__ . '/uploads/' . $newName);
}
```

> *«Вивернута» структура $_FILES для множинного завантаження — поширене джерело помилок у початківців: замість очікуваного масиву окремих файлів отримуємо масив окремих властивостей. Якщо потрібна зручніша структура (масив об'єктів «один файл — всі його властивості»), її легко зібрати самостійно циклом, як показано вище, звертаючись за спільним індексом $i до кожного під-масиву.*

Питання для самоперевірки

- Чому $_POST залишається порожнім, якщо клієнт надсилає JSON у тілі POST-запиту? Як прочитати такі дані?
- Для чого призначений php://input і чому його можна прочитати лише один раз за запит?
- Наведіть приклад, коли доцільно використати ім'я поля форми з [] (масив значень).
- Чим відрізняються валідація та санітизація вхідних даних? Наведіть по одному прикладу функції для кожної.
- Чому не можна довіряти значенню $_FILES['file']['type'] при перевірці типу завантаженого файлу?
- Для чого потрібна саме move_uploaded_file(), а не звичайний rename(), для переміщення завантаженого файлу?
- Опишіть, чим відрізняється структура $_FILES для одного файлу і для кількох файлів (name="photos[]").
- Що означає код помилки UPLOAD_ERR_INI_SIZE і як на нього вплинути з боку сервера?

Короткий глосарій

Термін	Означення
multipart/form-data	Формат кодування тіла запиту, що дозволяє передавати файли разом із текстовими полями
php://input	Потік для доступу до необробленого тіла HTTP-запиту
Валідація	Перевірка відповідності вхідних даних очікуваному формату чи правилам
Санітизація	Очищення чи екранування вхідних даних для безпечного використання
XSS (Cross-Site Scripting)	Вразливість, за якої зловмисний скрипт впроваджується через невекранований вивід
Тимчасовий файл завантаження	Файл, який PHP автоматично зберігає при отриманні завантаження до його переміщення
MIME-тип	Ідентифікатор формату вмісту файлу чи даних (напр. image/png)