



ТЕМА 6

Опрацювання HTTP-запитів і файлів

Дані форм: параметри та JSON · валідація вхідних даних · завантаження файлів на сервер

Бекенд-розробка, сервіси та мікросервіси

Комп'ютерна інженерія · 3 курс · Лекція 6 з 18

7 питань сьогоднішньої теми

01

HTTP-запити: методи та суперглобальні масиви

02

Дані форми у вигляді параметрів

03

Дані у форматі JSON

04

Валідація та фільтрація вхідних даних

05

Завантаження файлів: механізм і \$_FILES

06

Завантаження файлів: перевірка і збереження

07

Завантаження кількох файлів одночасно

Метод запиту та суперглобальні масиви

```
$method = $_SERVER['REQUEST_METHOD']; // 'GET', 'POST', 'PUT' ...  
  
if ($method === 'POST') {  
    // обробка даних форми  
}
```

Масив	Джерело даних
\$_GET	Параметри рядка запиту (?key=value)
\$_POST	Тіло запиту: form-urlencoded або multipart/form-data
\$_REQUEST	Об'єднання \$_GET, \$_POST, \$_COOKIE (рідко використовують)

\$_POST заповнюється лише для *form-urlencoded* і *multipart/form-data* — для *JSON*-тіла він порожній.

HTML-форма → \$_POST

HTML-форма

```
<form method="POST"
      action="submit.php">
  <input name="username">
  <input name="email">
  <button>Надіслати</button>
</form>
```

submit.php

```
$username = $_POST['username'] ?? '';
$email = $_POST['email'] ?? '';

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

Масиви полів: name="tags[]"

```
<input type="checkbox" name="tags[]" value="php">  
<input type="checkbox" name="tags[]" value="mysql">
```

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

Однйменні поля з [] автоматично збираються PHP в один масив — зручно для чекбоксів.

Читання JSON-тіла через php://input

```
// 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;
```

Формування JSON-відповіді

```
header('Content-Type: application/json');  
http_response_code(201);    // Created  
  
echo json_encode([  
    'status' => 'success',  
    'user' => ['id' => 42, 'name' => $name],  
]);
```

php://input можна прочитати лише один раз за запит — збережіть результат у змінну одразу.

filter_var() та типові фільтри

Фільтр	Призначення
FILTER_VALIDATE_EMAIL	Перевіряє формат e-mail
FILTER_VALIDATE_INT	Перевіряє й перетворює на ціле число
FILTER_VALIDATE_URL	Перевіряє формат URL

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

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

Санітизація: htmlspecialchars()

```
$comment = $_POST['comment'] ?? '';  
echo htmlspecialchars($comment, ENT_QUOTES, 'UTF-8');
```

XSS (Cross-Site Scripting): якщо вивести введений користувачем текст у HTML без екранування, зломисник може впровадити свій `<script>`. `htmlspecialchars()` перетворює `<`, `>`, `"`, `'` на безпечні відповідники.

Правило: ніколи не виводьте вхідні дані користувача в HTML без екранування.

Структура \$_FILES

```
<form method="POST" action="upload.php" enctype="multipart/form-data">
  <input type="file" name="avatar">
</form>
```

Ключ	Зміст
['name']	Оригінальне ім'я файлу на комп'ютері користувача
['type']	MIME-тип від браузера (не довіряти беззастережно)
['tmp_name']	Шлях до тимчасового файлу на сервері
['error']	Код помилки завантаження (0 — успіх)
['size']	Розмір файлу в байтах

Коди помилок завантаження

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

move_uploaded_file() з перевітками

```
$file = $_FILES['avatar'];

if ($file['error'] !== UPLOAD_ERR_OK) exit('Помилка');
if ($file['size'] > 2 * 1024 * 1024) exit('Завеликий файл');

$allowed = ['image/jpeg', 'image/png'];
$actualType = mime_content_type($file['tmp_name']);
if (!in_array($actualType, $allowed, true)) exit('Тип не дозволено');

$ext = pathinfo($file['name'], PATHINFO_EXTENSION);
$newName = uniqid('avatar_', true) . '.' . $ext;
$dest = __DIR__ . '/uploads/' . $newName;

move_uploaded_file($file['tmp_name'], $dest);
```

Дві критичні перестороги

['type'] ненадійний

Значення повідомляє браузер клієнта — його легко підробити. `mime_content_type()` перевіряє реальний вміст файлу на сервері.

Ім'я файлу від користувача — небезпечне

Може містити символи для обходу шляху (`../../etc/passwd`). Завжди генеруйте власне унікальне ім'я.

name="photos[]" — "вивернута" структура

```
// $_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;
    }
    $tmpName = $_FILES['photos']['tmp_name'][$i];
    $name = $_FILES['photos']['name'][$i];
    $ext = pathinfo($name, PATHINFO_EXTENSION);
    $newName = uniqid('photo_', true) . '.' . $ext;
    move_uploaded_file($tmpName, __DIR__ . '/uploads/' . $newName);
}
```

Підсумок теми 6

- `$_POST` заповнюється лише для `form-urlencoded/multipart` — для JSON читайте `php://input`
- Завжди перевіряйте (`filter_var`) і екрануйте (`htmlspecialchars`) вхідні дані
- `$_FILES` містить метадані завантаженого файлу; перевіряйте `['error']` першим
- `['type']` можна підробити — реальний тип перевіряє `mime_content_type()`
- `move_uploaded_file()` безпечніший за `rename()` для збереження завантажень
- Множинне завантаження (`photos[]`) дає "вивернуту" структуру масивів