

Upload plików, obsługa błędów i import CSV

Czas: 2 × 45 min

praca indywidualna

Cel: Bezpieczne przyjmowanie plików od użytkownika w PHP, czytelna obsługa i logowanie błędów oraz import danych z pliku CSV do bazy.

Po zajęciach potrafisz:

- buduje formularz wysyłania plików i odczytuje dane z `$_FILES` wraz z kodem błędu
- sprawdza rozmiar i rzeczywisty typ pliku (`fileinfo`), nadaje unikalną nazwę i zapisuje plik poza zasięgiem wykonywania skryptów
- tworzy miniatury biblioteką GD
- rozdziela wyświetlanie błędów (tryb deweloperski) od ich logowania do pliku
- importuje plik CSV do bazy z walidacją wierszy i raportem z importu

Sprzęt i materiały

- komputer z Windows 11, XAMPP, przeglądarka z narzędziami F12
- `r2_13_galeria.sql`, `r2_13_autorzy.csv`, foldery `r2_13_start` i `r2_13_testowe` z Pliki_do_zadan

Oprogramowanie

- **XAMPP** — PHP z rozszerzeniami `fileinfo` i `gd` (sprawdź w `phpinfo()`)
apachefriends.org/download.html
- **Visual Studio Code**
code.visualstudio.com

Przygotowanie przed zajęciami

- Przypomnij sobie z wykładu 2.10 budowę tablicy `$_FILES` i funkcję `move_uploaded_file()`.

Bezpieczeństwo

- Nigdy nie ufaj nazwie ani typowi pliku podanym przez przeglądarkę — sprawdzaj zawartość i nadawaj własną nazwę.
- Do testów używaj tylko plików z folderu `r2_13_testowe` lub własnych zdjęć bez danych osobowych (zdjęcia osób publikuje się tylko za ich zgodą).
- Plik logu może zawierać szczegóły techniczne — nie może być dostępny z przeglądarki.

Przebieg zadania

1 Projekt i formularz wysyłania

Skopiuj `r2_13_start` do `C:\xampp\htdocs\galeria`, zaimportuj `r2_13_galeria.sql`. W `phpinfo()` sprawdź, czy są sekcje **fileinfo** i **gd** — jeśli nie, w `C:\xampp\php\php.ini` usuń średnik przed `extension=fileinfo` / `extension=gd` i zrestartuj Apache. Zanotuj wartości `upload_max_filesize` i `post_max_size`.

1. W `dodaj.php` uzupełnij formularz: `enctype="multipart/form-data"`, ukryte pole `MAX_FILE_SIZE` (wartość z `MAX_ROZMIAR`) przed polem pliku, atrybuty `accept` i `required` w polu `zdjecie`.
2. Na początku obsługi POST wypisz tymczasowo `var_dump($_FILES)` i wyślij `gory.jpg`. Opisz w sprawozdaniu znaczenie pól `name`, `type`, `tmp_name`, `error`, `size`. Usuń `var_dump`.

2 Bezpieczne przyjęcie pliku

Błędy zbieraj w tablicy `$bledy`. Kolejność kontroli: tytuł → kod błędu → rozmiar → typ.

```
$typ = (new finfo(FILEINFO_MIME_TYPE))->file($plik['tmp_name']);  
$nazwa = bin2hex(random_bytes(16)) . '.' . DOZWOLONE_TYPY[$typ];
```

1. Napisz funkcję `komunikat_uploadu(int $kod): string` (instrukcja `match`), która zamienia kody `UPLOAD_ERR_INI_SIZE`, `UPLOAD_ERR_FORM_SIZE`, `UPLOAD_ERR_PARTIAL`, `UPLOAD_ERR_NO_FILE` i pozostałe na polskie komunikaty.
2. Sprawdź rozmiar (`MAX_ROZMIAR`) i typ funkcją `finfo` (ramka) — dozwolone tylko typy z `DOZWOLONE_TYPY`. Rozszerzenie zapisanego pliku bierz z tej tablicy, nie z nazwy od użytkownika.
3. Zapisz plik `move_uploaded_file()` pod losową nazwą w `uploads/`, a dane (tytuł, autor, nazwa, oryginalna nazwa, typ, rozmiar) w tabeli `zdjecia` zapytaniem przygotowanym. Po sukcesie przekieruj do `index.php?dodano=1`.
4. Napisz funkcję `utworz_miniature()`, która biblioteką GD (`imagecreatefromjpeg/png/webp`, `imagescale` do szerokości 400 px, `imagejpeg`) zapisze miniaturę w `uploads/min/`. Gdy GD nie ma (`function_exists('imagecreatetruecolor')`), funkcja zwraca `null`, a galeria pokazuje oryginał.
5. Przetestuj wszystkie pliki z `r2_13_testowe`. Oczekiwane: 3 zdjęcia przyjęte, `duzy_szum.png` — za duży, `notatka.jpg` i `prostokat.gif` — niedozwolony typ.

3 Galeria i ochrona katalogu

1. W `index.php` pobierz zdjęcia z imieniem i nazwiskiem autora (`LEFT JOIN`, najnowsze pierwsze) i wyświetl kafelki według wzoru w komentarzu: miniatura (`loading="lazy"`, `alt` = tytuł) z odnośnikiem do oryginału, tytuł, autor lub „autor nieznany”, rozmiar w KB. Wszystkie dane wypisuj przez `e()`.
2. Utwórz `uploads/.htaccess`, który wyłącza listowanie katalogu (`Options -Indexes`) i blokuje dostęp do plików z rozszerzeniami skryptów PHP (`<FilesMatch> + Require all denied`). Sprawdź, że `http://localhost/galeria/uploads/` zwraca 403.

4 Obsługa i logowanie błędów

Uzupełnij `config.php` w miejscu oznaczonym TODO.

1. Ustaw `error_reporting(E_ALL)`, `display_errors` zależne od `TRYB_DEV`, `log_errors` i `error_log = PLIK_LOGU`. Utwórz `logs/.htaccess` z `Require all denied`.
2. Napisz funkcję `loguj(string $poziom, string $tekst)` dopisującą do logu wiersz `[data czas] POZIOM: tekst` (`error_log($linia, 3, PLIK_LOGU)`). Loguj dodanie zdjęcia i każdy odrzucony plik.
3. Dodaj `set_exception_handler()`: wpis do logu + kod 500 + w trybie DEV szczegóły, w produkcji tylko „Przepraszamy, wystąpił błąd”. Test: zmień w `db.php` nazwę bazy na błędną, otwórz galerię przy `TRYB_DEV = true` i `false`, obejrzyj log, przywróć poprawną nazwę.

5 Import autorów z CSV

Otwórz `r2_13_autorzy.csv` w Notatniku. Formularz w `import.php` przyjmuje plik CSV (`$_FILES['csv']`). Czytaj go wywołaniem `fgetcsv($f, null, ';', '"', '')`.

1. Sprawdź nagłówek (`imie;nazwisko;klasa`), usuwając z niego znak BOM (`preg_replace('/^\xEF\xBB\xBF/', '', ...)`).
2. Każdy wiersz zwaliduj: 3 kolumny, niepuste imię i nazwisko, klasa zgodna z `/^[1-5]T[IP]$/`. Poprawne wstaw przez `INSERT IGNORE` (w tabeli jest UNIQUE na imię, nazwisko i klasę), całość w transakcji.
3. Pokaż raport: dodane, pominięte (już istnieją — `rowCount() = 0`), błędne z numerem wiersza i powodem; zapisz go też w logu. Zaimportuj plik dwa razy i porównaj raporty. Pod formularzem wyświetl tabelę autorów.
4. Zrzuty: komunikaty błędów uploadu, galeria, 403 dla `uploads/`, oba raporty importu, fragment `app.log`. Spakuj projekt (bez folderu `uploads`) jako `nazwisko_imie_zad13.zip`.

W sprawozdaniu umieść

- ☐ tabela wykonanych poleceń (krok.punkt) z kluczowymi fragmentami kodu
- ☐ opis pól `$_FILES` i wartości `upload_max_filesize` / `post_max_size`
- ☐ zrzuty: błędy uploadu, galeria, 403 dla katalogu, raporty importu, fragment logu
- ☐ archiwum `nazwisko_imie_zad13.zip`
- ☐ odpowiedzi na pytania kontrolne

Pytania kontrolne

1. Dlaczego nie można polegać na `$_FILES['zdjecie']['type']` ani na rozszerzeniu pliku?
2. Po co serwer nadaje plikom własne, losowe nazwy?
3. Dlaczego na serwerze produkcyjnym `display_errors` musi być wyłączone, a błędy i tak powinny być zapisywane?
4. Co się stanie z `$_POST` i `$_FILES`, gdy wysłany plik przekroczy `post_max_size`? Jak to wykryć?
5. Dlaczego import CSV warto wykonać w transakcji?

Kryteria oceny

Kryterium	Punkty
Formularz z enctype i MAX_FILE_SIZE, opis <code>\$_FILES</code>	2
Kontrola kodu błędu, rozmiaru i typu (finfo); 3 pliki odrzucone z właściwym komunikatem	4
Losowa nazwa, zapis do bazy, miniatura GD	3
Galeria z escapowaniem, .htaccess w <code>uploads/</code> (403)	2
Konfiguracja błędów, funkcja <code>loguj()</code> , obsługa wyjątków w DEV i produkcji	3
Import CSV: BOM, walidacja, INSERT IGNORE w transakcji, raport 8/1/3	4
Zrzuty, archiwum i odpowiedzi na pytania	2
Razem	20

Sprawozdanie przygotuj na wzorze dołączonym do zadania (plik .docx o tym samym numerze).