

PHP — obsługa formularzy

Czas: 2 × 45 min

praca indywidualna

Cel: Odbieranie danych z formularzy metodami GET i POST, walidacja po stronie serwera z komunikatami przy polach, ponowne wypełnianie pól z bezpiecznym wypisywaniem (`htmlspecialchars`) oraz zastosowanie wzorca Post/Redirect/Get.

Po zajęciach potrafisz:

- dobiera metodę GET lub POST do rodzaju formularza
- sprawdza dane funkcjami `isset`, `filter_input`, `filter_var` z zakresami
- wyświetla komunikaty błędów przy polach i zachowuje wpisane wartości
- wypisuje dane użytkownika przez `htmlspecialchars`
- stosuje wzorzec PRG (`header('Location: ...')` i `exit`)

Sprzęt i materiały

- komputer z Windows 11 i XAMPP
- folder `r2_07_start` z Pliki_do_zadan (`styl.css`, `podroz.html`, `rata.html`)

Oprogramowanie

- **XAMPP (Apache, PHP 8, MariaDB, phpMyAdmin)** — projekt w folderze `C:\xampp\htdocs`
apachefriends.org/download.html
- **Visual Studio Code**
code.visualstudio.com
- **Przeglądarka Chrome, Edge lub Firefox** — narzędzia deweloperskie F12 (Elementy, Konsola, Sieć, Aplikacja)

Przygotowanie przed zajęciami

- Przeglądnij wykład 2.06 (`$_GET`, `$_POST`, `filter_var`, PRG).

Bezpieczeństwo

- Każdą wartość od użytkownika wypisuj przez funkcję `e()` (`htmlspecialchars` z `ENT_QUOTES`) — także w atrybucie `value`.
- Plik `zapytania.csv` zawiera dane osobowe (imię, e-mail) — używaj tylko danych testowych i usuń plik po zajęciach. Na prawdziwym serwerze taki plik nie może leżeć w katalogu dostępnym z przeglądarki.

Przebieg zadania

1 Funkcje pomocnicze

Skopiuj pliki do `C:\xampp\htdocs\nazwisko_z07`. Zmień nazwy `podroz.html` → `podroz.php`, `rata.html` → `rata.php`. Utwórz `funkcje.php`.

```
function e(?string $tekst): string {
    return htmlspecialchars($tekst ?? '', ENT_QUOTES, 'UTF-8');
}
// liczba("6,5", 2, 30) → 6.5;  liczba("abc", 2, 30) → null
function liczba(?string $wartosc, float $min, float $max): ?float { ... }
```

- Przepisz funkcję `e()` i napisz `liczba()`: zamień przecinek na kropkę, sprawdź `filter_var(..., FILTER_VALIDATE_FLOAT, ['options' => ['min_range' => $min, 'max_range' => $max]])`, zwróć `null`, gdy wynik to `false`.
- Napisz `zl(float $kwota): string` oraz `bładPola(array $bledy, string $pole): string`, która zwraca `<p class="bład">...</p>` albo pusty tekst.

2 Koszt podróży — GET

Formularz GET nadaje się do obliczeń, które nic nie zmieniają na serwerze: wynik można zapisać jako zakładkę lub wystać linkiem.

- Formularz uznaj za wysłany, gdy `isset($_GET['dystans'])`. Sprawdź: dystans 1–5000 km, spalanie 2–30 l/100 km, cena 3–15 zł/l (wszystkie z przecinkiem lub kropką), liczba osób całkowita 1–9 — przez `filter_input(INPUT_GET, 'osoby', FILTER_VALIDATE_INT, ...)`.
- Błędy zbieraj w tablicy `$bledy['pole'] = 'komunikat'`; wyświetl je pod polami, a błędne pola oznacz klasą `zle`. Po wysłaniu pola zachowują wpisane wartości, pole „W obie strony” — zaznaczenie.
- Gdy dane są poprawne, oblicz: liczbę km (×2 dla podróży w obie strony), litry, koszt i koszt na osobę. Test: 320 km, 6,5 l, 6,29 zł, 3 osoby, w obie strony. Skopiuj adres z paska i otwórz w nowej karcie — co się stało?

3 Rata kredytu — POST

Rata równa: $R = K \cdot r / (1 - (1 + r)^{-n})$, gdzie K — kwota, r — oprocentowanie roczne / 100 / 12, n — liczba rat. Dla $r = 0$: $R = K / n$.

- Obsłuż formularz, gdy `$_SERVER['REQUEST_METHOD'] === 'POST'`. Sprawdź: kwota 1000–500 000 zł, oprocentowanie 0–30%, liczba rat całkowita 6–360. Pokaż ratę, łączną kwotę do spłaty i odsetki.
- Wpisz w pole kwoty tekst z cudzysłowem, np. `30 000 "zł"`. Obejrzyj źródło strony (Ctrl+U) i zapisz, jak wygląda atrybut `value`.
- Test: 30 000 zł, 8,5%, 36 rat; drugi test: oprocentowanie 0%.

4 Zapytanie do doradcy i PRG

Formularz ma dwa przyciski `submit` z `name="akcja"` i wartościami `oblicz` oraz `zapytanie`.

```
$plik = fopen(__DIR__ . '/zapytania.csv', 'a');
fputcsv($plik, $wiersz, ';');
fclose($plik);
header('Location: rata.php?wyslano=1', true, 303);
exit;
```

1. Dla akcji `zapytanie` sprawdź dodatkowo imię (min. 3 znaki, `mb_strlen`) i e-mail (`FILTER_VALIDATE_EMAIL`). Jeśli wszystko jest poprawne, dopisz wiersz do `zapytania.csv` (data, imię, e-mail, kwota, procent, raty, rata) i przekieruj przeglądarkę (kod powyżej).
2. Gdy w adresie jest `wyslano=1`, pokaż komunikat „Dziękujemy! Zapytanie zostało zapisane”. Odśwież stronę (F5) — czy wiersz w CSV się powtórzył? Usuń na chwilę `header` i `exit`, wyślij zapytanie, naciśnij F5 i opisz różnicę. Przywróć PRG.
3. W F12 → Sieć obejrzyj żądanie POST: kod odpowiedzi 303, nagłówek `Location` i dane formularza (Ładunek/Payload). Zrób zrzuty i spakuj folder (bez pliku CSV).

W sprawozdaniu umieść

- ☐ tabela testów obu kalkulatorów (dane wejściowe, oczekiwany i otrzymany wynik)
- ☐ adres URL z wynikiem kalkulatora podróży i wyjaśnienie, dlaczego działa w nowej karcie
- ☐ fragment źródła strony z atrybutem `value` dla tekstu z cudzysłowem
- ☐ opis zachowania F5 z PRG i bez PRG oraz zrzut zakładki Sieć (303, Location)
- ☐ spakowany folder `nazwisko_z07`

Pytania kontrolne

1. Kiedy wybrać GET, a kiedy POST? Podaj po jednym przykładzie z tego zadania.
2. Dlaczego walidacja na serwerze jest konieczna, nawet gdy formularz ma atrybuty HTML i walidację w JS?
3. Co robi `htmlspecialchars` i dlaczego potrzebna jest flaga `ENT_QUOTES`?
4. Jaki problem rozwiązuje wzorzec PRG i dlaczego po `header('Location: ...')` piszemy `exit`?

Kryteria oceny

Kryterium	Punkty
Funkcje <code>e()</code> , <code>liczba()</code> , <code>zł()</code> , <code>bladPola()</code>	3
Kalkulator GET: 4 reguły walidacji, komunikaty, zachowane wartości, poprawny wynik	5
Kalkulator POST: walidacja, wzór raty (także 0%), odsetki, <code>htmlspecialchars</code> w <code>value</code>	5
Zapytanie: walidacja imienia i e-maila, zapis CSV, PRG z komunikatem	4
Testy, analiza F5 i zakładki Sieć, odpowiedzi na pytania	3
Razem	20

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