JAK DZIAŁAJĄ FORMULARZE

Odczyt danych metodami POST i GET

Metody przesyłania

POST

```
<form action="" method="POST">…
```

Dane przesyłane w treści żądania HTTP, nie są widoczne w URL-u.

METODA UKRYTA

Dane zapisują się w tablicy superglobalnej $_POST

GET

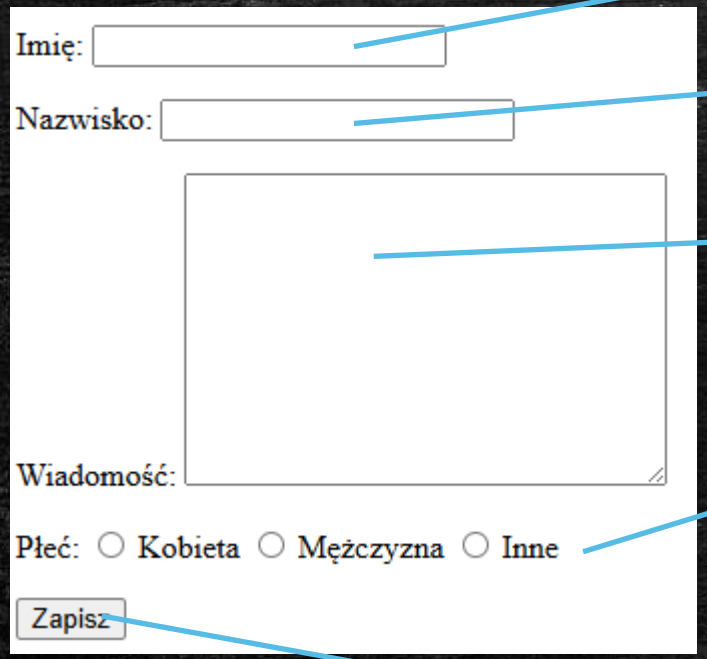
```
<form action="" method="GET">…
```

Dane są dołączane do URL-a w formacie *?klucz=wartosc&klucz2=...* .

METODA JAWNA

Dane zapisują się w tablicy superglobalnej $_GET

Budowa prostego formularza




```
<form action="" method="GET">
    Imię: <input type="text" name="imie" />

    Nazwisko: <input type="text" name="nazwisko" />

    Wiadomość: <textarea name="wiadomosc" cols="30"
    rows="10"></textarea>

    Płeć:
    <input type="radio" name="plec" value="k"> Kobieta
    <input type="radio" name="plec" value="m"> Mężczyzna
    <input type="radio" name="plec" value="i"> Inne


    <input type="submit" value="Zapisz" name="przycisk" />

</form>
```

Znaczenie *name*

```
<form action="" method="GET">
    Imię: <input type="text" name="imie" />

    Nazwisko: <input type="text" name="nazwisko" />

    Wiadomość: <textarea name="wiadomosc" cols="30"
    rows="10"></textarea>

    Płeć:
    <input type="radio" name="plec" value="k"> Kobieta
    <input type="radio" name="plec" value="m"> Mężczyzna
    <input type="radio" name="plec" value="i"> Inne

    <input type="submit" value="Zapisz" name="przycisk" />

</form>
```

name pozwala na identyfikację elementów w tablicy superglobalnej $_POST lub $_GET

podane nazwy stają się indeksami w tablicy

I gdzie te dane? - jak wygląda tablica $_GET

```html
<form action="" method="GET">
    <p>Imię: <input type="text" name="imie" /></p>
    <p>Nazwisko: <input type="text" name="nazwisko" /></p>
    <p>Wiadomość: <textarea name="wiadomosc" cols="30" rows="10
    <p>Płeć:
        <input type="radio" name="plec" value="k"> Kobieta
        <input type="radio" name="plec" value="m"> Mężczyzna
        <input type="radio" name="plec" value="i"> Inne
    </p>
    <input type="submit" value="Zapisz" name="przycisk" />
</form>
```

```
Array
(
    [imie] => Jan
    [nazwisko] => Kowalski
    [wiadomosc] => treść wiadomości
    [plec] => m
    [przycisk] => Zapisz
)
```

A tak to wygląda w URL'u

localhost/zajecia/formularze/prosty/form_simple.php?imie=Jan&nazwisko=Kowalski&wiadomosc=treść+wiadomości&plec=m&przycisk=Zapisz

Jak podejrzeć tablicę superglobalną

```php
<?php
echo "<pre>";
print_r(value: $_GET);
echo "</pre>";
?>
```

```
Array
(
    [imie] => Jan
    [nazwisko] => Kowalski
    [wiadomosc] => treść wiadomości
    [plec] => m
    [przycisk] => Zapisz
)
```

```php
<?php
echo "<pre>";
var_dump(value: $_GET);
echo "</pre>";
?>
```

```
array(5) {
  ["imie"]=>
  string(3) "Jan"
  ["nazwisko"]=>
  string(8) "Kowalski"
  ["wiadomosc"]=>
  string(19) "treść wiadomości"
  ["plec"]=>
  string(1) "m"
  ["przycisk"]=>
  string(6) "Zapisz"
}
```

Użycie znaczników <pre></pre> pozwala na czytelniejsze wyświetlenie tablicy, bez nich wynik wyświetlony zostanie w jednej linii

Wyświetlanie elementów tablicy $_GET lub $_POST

$_POST

```php
<?php
echo $_POST['nazwa_indeksu’];
?>
np. echo $_POST['imie'];
```

$_GET

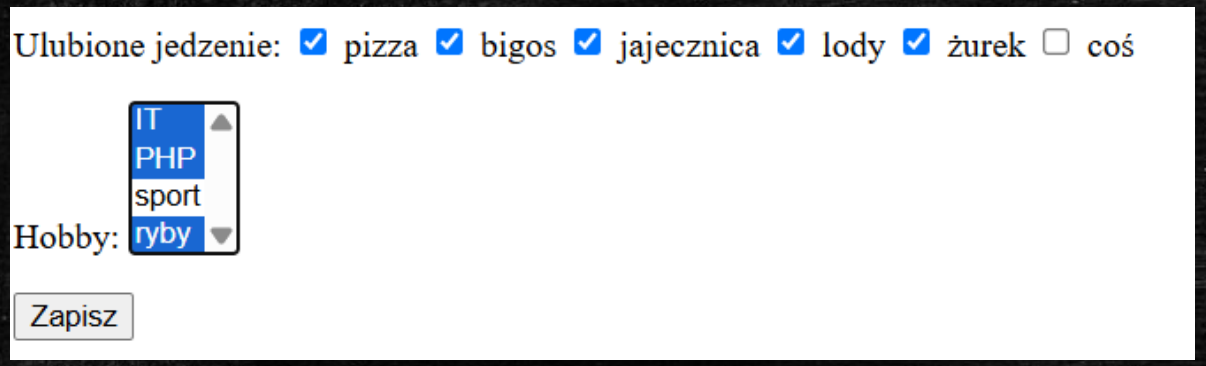
```php
<?php
echo $_GET['nazwa_indeksu'];
?>
np. echo $_GET['nazwisko'];
```

nazwa tablicy
(z góry określona)

```
Array
(
    [imie] => Jan
    [nazwisko] => Kowalski
    [wiadomosc] => treść wiadomości
    [plec] => m
    [przycisk] => Zapisz
)
```

Co z wielokrotnym wyborem? checkbox i select multiple

```
<p>Ulubione jedzenie:
    <input type="checkbox" name="food[]" value="pizza"> pizza
    <input type="checkbox" name="food[]" value="bigos"> bigos
    <input type="checkbox" name="food[]" value="jajecznica"> jajecznica
    <input type="checkbox" name="food[]" value="lody"> lody
    <input type="checkbox" name="food[]" value="zurek"> żurek
    <input type="checkbox" name="food[]" value="cos"> coś
</p>
<p>Hobby:
    <select name="hobby[]" multiple>
        <option value="IT">IT</option>
        <option value="PHP">PHP</option>
        <option value="sport">sport</option>
        <option value="ryby">ryby</option>
    </select>
</p>
```



Jeżeli mamy możliwość zaznaczenia wielu opcji – wrzucamy je do tablicy *name = " nazwa_tablicy[ ]"*

```
Array
(
    [food] => Array
        (
            [0] => pizza
            [1] => bigos
            [2] => jajecznica
            [3] => lody
            [4] => zurek
        )

    [hobby] => Array
        (
            [0] => IT
            [1] => PHP
            [2] => ryby
        )

    [przycisk] => Zapisz
)
```

checkbox i select multiple - wyświetlanie

```
Array
(
    [food] => Array
        (
            [0] => pizza
            [1] => bigos
            [2] => jajecznica
            [3] => lody
            [4] => zurek
        )

    [hobby] => Array
        (
            [0] => IT
            [1] => PHP
            [2] => ryby
        )

    [przycisk] => Zapisz
)
```

Wyświetlamy pojedynczy element:

```php
<?php
echo $_GET['indeks1']['indeks2'];
?>
```

np. `echo $_GET['food'][0];`

wyświetli *pizza*

Wyświetlamy wszystkie zaznaczone opcje:

```php
<?php
foreach($_GET['hobby'] as $hobby){
   echo $hobby."<br>";
}
?>
```

IT
PHP
ryby

Sprawdzanie, czy formularz został przesłany

Możemy sprawdzić czy przesłano formularz

```php
<?php
    if ($_SERVER['REQUEST_METHOD'] === 'POST') {
        echo "Formularz został wysłany!";
    }
?>
```

lub możemy sprawdzić czy naciśnięto przycisk

```html
<input type="submit" value="Zapisz" name="przycisk" />
```

```php
<?php
    if(isset($_POST['przycisk'])) {
        echo "Naciśnięto przycisk";
    }
?>
```

Nie pracujemy na nieistniejących danych, czyli sprawdzamy czy pola zostały uzupełnione

isset()

Sprawdza czy zmienna została przesłana i nie jest NULL

```
$zmienna='';
echo isset($zmienna)
      ? 'zmienna została przesłana i nie jest NULL'
        : 'zmienna nie została przesłana lub jest
NULL';
```

Nie pracujemy na nieistniejących danych, czyli sprawdzamy czy pola zostały uzupełnione

empty()

Ustala czy zmienna jest pusta.

Następujące wartości są uważane za puste:
'' (pusty ciąg)
0 (0 jako integer)
0.0 (0 jako float)
'0' (0 jako string)
NULL
FALSE
array() (pusta tablica)

```php
$zmienna=0;
if (empty($zmienna)) { echo '$zmienna jest równa zero,
pusta lub w ogóle nie została ustalona';}
```

przy polach, które mogą mieć wartość „0”, lepiej użyć:

```php
if ($wiek === '' || $wiek === null) {
    echo "Wiek jest wymagany";
}
lub
if (!isset($_POST['wiek']) || $_POST['wiek']
=== '') {
    echo "Pole wiek nie może być puste";
}
```

empty(), isset() przykład

```php
<?php
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    // pobieramy wartości pól
    $imie = $_POST['imie'] ?? '';
    $wiek = $_POST['wiek'] ?? '';

    // Sprawdzenie, czy pola istnieją
    if (!isset($_POST['imie'], $_POST['wiek'])) {
        echo "Brakuje wymaganych pól!";
    }

    // Sprawdzenie, czy pola nie są puste
    if (empty($imie) || empty($wiek)) {
        echo "Wszystkie pola są obowiązkowe!";
    } else {
        echo "Imię: $imie, Wiek: $wiek";
    }
}
?>
```

Filtrowanie i zabezpieczenia pól przed atakami

Najczęstsze zagrożenia:

- **XSS** (Cross-Site Scripting) → np. ktoś wpisze <script>...</script> w polu formularza
- **Injection** → próby manipulacji zapytaniami lub danymi
- **Fałszywe dane** (np. zmienione przez DevTools)

Dlatego stosujemy **kilka warstw zabezpieczeń**

htmlspecialchars() - ochrona przed XSS

Każde dane, które **wyświetlasz z powrotem w HTML**, należy przepuścić przez tę funkcję:

```
$bezpieczne_imie = htmlspecialchars($imie, ENT_QUOTES, 'UTF-8');
echo "Witaj, $bezpieczne_imie!";
```

Używaj **zawsze przy wyświetlaniu**, nie przy zapisie

Zamienia <, >, ", ' na bezpieczne encje HTML

filter_input() - pobieranie danych z walidacją

PHP ma wbudowany system filtrów

```php
$imie = filter_input(INPUT_POST, 'imie',
FILTER_SANITIZE_SPECIAL_CHARS);
$wiek = filter_input(INPUT_POST, 'wiek', FILTER_VALIDATE_INT);

if ($wiek === false) {
    echo "Wiek musi być liczbą!";
}
```

Łatwiejsze i czytelniejsze niż ręczne $_POST

Możliwość walidacji e-maili, URL-i, liczb itd

Walidacja długości i formatu

Dodatkowo warto sprawdzać długości pól i dopuszczalne znaki

```php
if (strlen($imie) < 2 || strlen($imie) > 50) {
    echo "Imię musi mieć od 2 do 50 znaków.";
}

if (!preg_match('/^[a-ząćęłńóśźżA-ZĄĆĘŁŃÓŚŹŻ\s-]+$/', $imie)) {
    echo "Imię zawiera niedozwolone znaki.";
}
```

Tokeny CSRF

Przy poważniejszych formularzach warto też stosować **tokeny CSRF**, żeby zabezpieczyć się przed automatycznym wysyłaniem formularzy przez złośliwe strony

```php
<?php
// przy wyświetlaniu formularza
session_start();
$_SESSION['token'] = bin2hex(random_bytes(32));
?>
<form method="post">
  <input type="hidden" name="token" value="<?= $_SESSION['token'] ?>">
  <input type="text" name="imie">
  <button type="submit">Wyślij</button>
</form>
<?php
// przy przetwarzaniu
if (!hash_equals($_SESSION['token'], $_POST['token'] ?? '')) {
    die('Błąd CSRF!');
}?>
```

Podsumowanie

Co sprawdzamy	Jak to zrobić
Czy formularz wysłany	$_SERVER['REQUEST_METHOD'] === 'POST'
Czy pola istnieją	isset() lub ?? ''
Czy pola wypełnione	$_POST['x'] === '' (lepiej niż empty() dla 0)
Ochrona XSS	htmlspecialchars() przy wyświetlaniu
Walidacja danych	filter_input(), regex, strlen
Ochrona przed CSRF	token w sesji i formularzu
Inne zabezpieczenia	Limity długości, typów, znaków

Kompletny przykład formularza z walidacją i zabezpieczeniami

```php
<?php
session_start();
if (empty($_SESSION['token'])) {
    $_SESSION['token'] = bin2hex(string: random_bytes(bytes: 32));
}

$errors = [];
$imie = '';
$wiek = '';

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    // CSRF
    if (!hash_equals(a: $_SESSION['token'], b: $_POST['token'] ?? '')) {
        die('Nieprawidłowy token CSRF!');
    }

    // Pobranie danych z filtrowaniem
    $imie = filter_input(type: INPUT_POST, var_name: 'imie', filter: FILTER_SANITIZE_SPECIAL_CHARS);
    $wiek = $_POST['wiek'] ?? '';

    // Walidacja
    if ($imie === '' || strlen(string: $imie) < 2) {
        $errors[] = "Imię jest wymagane i musi mieć min. 2 znaki.";
    }

    if ($wiek === '' && $wiek !== '0') {   // uwzględniamy '0'
        $errors[] = "Wiek jest wymagany.";
    } elseif (!ctype_digit(text: $wiek)) {
        $errors[] = "Wiek musi być liczbą całkowitą.";
    }

    if (empty($errors)) {
        echo "<p>Dane poprawne:</p>";
        echo "Imię: " . htmlspecialchars(string: $imie) . "<br>";
        echo "Wiek: " . htmlspecialchars(string: $wiek);
    }
}
?>

<form method="post">
  <input type="hidden" name="token" value="<?= $_SESSION['token'] ?>">
  Imię: <input type="text" name="imie" value="<?= htmlspecialchars(string: $imie) ?>"><br>
  Wiek: <input type="text" name="wiek" value="<?= htmlspecialchars(string: $wiek) ?>"><br>
  <button type="submit">Wyślij</button>
</form>

<?php
if (!empty($errors)) {
    echo "<ul>";
    foreach ($errors as $e) echo "<li>$e</li>";
    echo "</ul>";
}
?>
```