Obsługa formularzy – metody POST i GET

Zajmijmy się teraz formularzami, czyli jednym z najważniejszych elementów komunikacji z użytkownikiem na stronach internetowych. Jeśli kiedykolwiek zamawialiśmy coś w sklepie internetowym, rejestrowaliśmy się na stronie lub logowaliśmy się do systemu, to na pewno korzystaliśmy z formularzy. Dzięki nim użytkownik może wprowadzać dane, które następnie przesyłamy do serwera i przetwarzamy w PHP.

Pora dobrze zrozumieć różnice między obsługą metod

`GET` oraz `POST`. Poznamy też dwa sposoby sprawdzania, czy formularz został w ogóle wysłany:

- Wykorzystanie funkcji `isset($_POST["name"])`
- Sprawdzenie metody w globalnej tablicy `$_SERVER["REQUEST_METHOD"]`

Jak działa formularz – metoda GET lub POST

Formularz w HTML to zestaw pól, w które użytkownik wpisuje dane, a po kliknięciu przycisk “Wyślij”, przeglądarka wysyła te informacje do serwera – tworzenie interfejsów formularzy za pomocą klasycznych kontrolek opisaliśmy to dokładnie w lekcji: Formularz oraz kontrolki formularza.

Natomiast w PHP obsługujemy formularze, czyli dokonujemy przetworzenia tych podanych informacji – np. rezerwujemy produkt w sklepie czy postujemy czyjś komentarz na stronie.

Przetworzenia przesłanych danych dokonujemy w PHP za pomocą dwóch metod:

- **GET** – dane są przekazywane w adresie URL.
- **POST** – dane są wysyłane w tle, bez doklejania ich do adresu URL.

Zobaczmy krótkie podsumowanie różnic w postaci tabelarycznej:

Cecha	GET	POST
Dane widoczne w adresie URL?	Tak	Nie
Nadaje się do formularzy logowania?	Nie	Tak
Można zakładać, że użytkownik nie zmodyfikuje danych?	Nie	Tak
Nadaje się do wyników wyszukiwania?	Tak	Nie
Maksymalna ilość danych?	Ograniczona	Brak limitu

Odbieranie danych z formularza w PHP

Wartość podaną przez użytkownika w formularzu możemy odebrać w skrypcie PHP w specjalnych, globalnie dostępnych tablicach asocjacyjnych:

`$_POST` lub `$_GET`. Oczywiście wszystko zależy od metody przesyłu danych ustawionej w formularzu.

Odczyt danych po stronie PHP

```
<?php
    // Odczyt, jeśli wybrano dla formularza metodę POST
    $wartosc = $_POST["nazwa_pola"];
    // Odczyt, jeśli wybrano metodę GET
    $wartosc = $_GET["nazwa_pola"];
?>
```

Atrybut

`name` w kontrolce formularza określa nazwę indeksu, pod którą wartość zostanie umieszczona w tablicy `$_POST` lub `$_GET`. Atrybut `id` służy najczęściej do powiązań z JavaScriptem lub z etykietą `<label>` w HTML. W praktyce często używamy obu: `name` do przesyłu danych, `id` do logiki interfejsu.

Tyle szablon odczytu. Jednak nie możemy tak od razu dokonać w PHP zwykłego “zczytania” podanych przez internautę wartości. Dlaczego? Ponieważ być może nie zdążył on jeszcze niczego do nas wysłać, a jednie zobaczył istnienie formularza w przeglądarce!

Jak sprawdzić, czy formularz został wysłany?

To pierwsza, najważniejsza sprawa – otóż jeszzce przed zrealizowaniem odczytu wartości trzeba najpierw dokonać weryfikacji przesłania formularza, ponieważ już pierwsze wejście użytkownika na stronę może zakończyć się ostrzeżeniem. W końcu dane nie będą istnieć, ponieważ użytkownik nie miał jeszcze możliwości do nas nic wysłać! Zobaczymy wówczas ostrzeżenie:

```
Notice: Undefined index: nazwa_pola
```

Dlatego przed odczytaniem wartości należy upewnić się, że formularz faktycznie został przesłany. Można to zrobić na dwa sposoby:

1. Sprawdzenie funkcją isset()

```
<?php
if (isset($_POST["nazwa_pola"])) {
    // wartość została przesłana – można przetwarzać
}
?>
```

Metoda ta sprawdza, czy dana wartość w tablicy została ustawiona. Nie mylmy jej z funkcją

`empty()` – ta druga służy do sprawdzenia, czy wartość nie jest pusta, ale *zakłada, że indeks istnieje*.

2. Sprawdzenie metody żądania HTTP

```
<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // formularz przesłano metodą POST
}
?>
```

To bardziej ogólne podejście – sprawdzamy, czy użytkownik przesłał dane za pomocą żądania typu POST. Uwaga: ta metoda nie działa w przypadku formularzy wysyłanych metodą GET – ponieważ każde zwykłe wejście na stronę również jest żądaniem GET. Dopiero posłanie przez internauty danych metodą POST jest – jak to często nieprzypadkowo mówimy – "postowaniem" informacji na serwerze.

Przykład odczytu danych – metoda POST

Załóżmy, że chcemy zebrać imię użytkownika i powitać go na stronie. Użyjemy metody POST – dane nie będą widoczne w adresie URL.

```
<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    if (!empty($_POST["imie"])) {
        $imie = htmlspecialchars($_POST["imie"]);
        echo "<p>Witaj, $imie!</p>";
    } else {
        echo "<p>Nie podano imienia.</p>";
    }
}
?>
<form method="post">
    <label for="imie">Podaj imię:</label>
    <input type="text" name="imie" id="imie">
    <button type="submit">Wyślij</button>
</form>
```

Analiza co dzieje się w tym kodzie:

- Na początku sprawdzamy, czy formularz został przesłany metodą `POST` – tak jak to opisywaliśmy.
- Jeśli coś "zapostowano" – odczytujemy wartość pola `imie` z tablicy `$_POST`.
- Funkcja `htmlspecialchars()` zabezpiecza przed wstrzyknięciem tagów HTML chroni nas przed wstrzyknięciem tagów HTML do zmiennych, zamieniając znaki tworzące tagi na encje.
- Jeśli użytkownik nie wypełni pola, pokażemy komunikat, że dane są puste.

Przykład odczytu danych z formularza – metoda GET

W tej wersji stworzymy formularz wyszukiwania – użytkownik wpisze nazwę filmu, a my wypiszemy, co chciał wyszukać. Tym razem użyjemy metody GET – dane pojawią się w adresie URL.

```
<?php
if (isset($_GET["tytul"])) {
    $tytul = htmlspecialchars($_GET["tytul"]);
    if (!empty($tytul)) {
        echo "<p>Szukasz filmu: <strong>$tytul</strong></p>";
    } else {
        echo "<p>Nie podano tytułu filmu.</p>";
    }
}
?>
<form method="get">
    <label for="tytul">Tytuł filmu:</label>
```

```
    <input type="text" name="tytul" id="tytul">
    <button type="submit">Szukaj</button>
</form>
```

Analiza powyższego kodu:

- Formularz używa metody `GET` – po wysłaniu dane pojawią się w adresie – na przykład: `?tytul=Avatar`.
- Dzięki `isset()` oraz `!empty()` sprawdzamy, czy dane zostały rzeczywiście przesłane i nie są puste.
- Funkcja `htmlspecialchars()` również tutaj chroni nas przed wstrzyknięciem tagów HTML do zmiennych, zamieniając znaki tworzące tagi na encje.

Podsumowanie wiedzy z lekcji

Formularze HTML to jeden z najczęstszych sposobów przesyłania danych od użytkownika do skryptu PHP. Aby dane te odebrać i przetworzyć, musimy znać różnice między dwoma podstawowymi metodami przesyłu:

`GET` i `POST:`

Metoda GET:

- Dane widoczne są w adresie URL.
- Nadaje się np. do wyszukiwarek.
- Ograniczona ilość przesyłanych danych.
- Nie nadaje się do haseł ani danych poufnych.

Metoda POST:

- Dane przesyłane są w tle – niewidoczne w adresie, choć też domyślnie niezaszyfrowane (dopiero użycie HTTPS zapewnia szyfrowanie).
- Bezpieczniejsza metoda, bo nie ujawnia wartości “wpost” – używana np. w logowaniu, rejestracji.
- Brak limitu długości danych.

W PHP dane z formularza odbieramy przez specjalne tablice globalne:

- `$_POST` – dla metody POST
- `$_GET` – dla metody GET

Koniecznie trzeba najpierw sprawdzić, czy formularz został przesłany:

- Funkcją `isset()`
- Lub przez warunek `$_SERVER["REQUEST_METHOD"] == "POST"`

Dobre praktyki:

- Zabezpieczaj dane funkcją `htmlspecialchars()`, by uniknąć wstrzyknięć tagów HTML – zamieńmy wrażliwe nawiasy ostre na encje.
- Sprawdzaj, czy dane nie są puste – używaj `!empty()`.
- Dopiero po walidacji wykonuj przetwarzanie – np. wyświetlanie wartości czy zapisywanie ich do bazy.

Dzięki tej lekcji potrafisz już:

- Odczytywać dane z formularza metodą GET lub POST,
- Odróżniać te metody i wybierać odpowiednią do sytuacji,
- Weryfikować, czy formularz został przesłany oraz zabezpieczyć dane wejściowe przed wstrzyknięciem tagów HTML.

Oczywiście w przypadku ewentualnej współpracy z bazą danych kolejnym niebezpieczeństwem przy przyjmowaniu danych od użytkownika metodą POST jest atak wstrzykiwania SQL – więcej na ten temat dowiemy się w tym miejscu. Na egzaminach INF.03 raczej nie realizujemy ochrony przed atakiem tego typu, lecz oczywiście warto znać i rozumieć jego naturę.