Instrukcje warunkowe w PHP

W każdym języku programowania prędzej czy później musimy – jak to się mówi – “rozgałęzić” działanie naszego skryptu, czyli wykonać konkretne instrukcje tylko wtedy, gdy spełniony zostanie (lub wręcz przeciwnie) określony warunek logiczny. W języku PHP (podobnie jak to było w JavaScript) służy do tego klasyczna instrukcja warunkowa

`if`. I właśnie nią teraz się zajmijmy – tym bardziej, że niesamowicie często tego typu instrukcje trzeba poprawnie implementować rozwiązując zadania praktyczne INF.03.

Ponieważ w lekcji: Instrukcje warunkowe w JS wyjaśniliśmy już wiele fundamentalnych zasad logicznych dotyczących ifów, przedstawiliśmy też tzw. spójniki logiczne oraz ich hierarchię, to gorąco zalecam **zapoznać się najpierw** właśnie z tamtą lekcją z rozdziału o JavaScript. Natomiast niniejszy artykuł – o instrukcjach warunkowych w PHP – potraktujmy bardziej jako przedstawienie różnic składniowych pomiędzy PHP a JavaScript – nie będziemy zatem po raz drugi bardzo szczegółowo wyjaśniać tych samych zasad logicznych i operatorów.

Składnia instrukcji warunkowej w PHP

Instrukcja warunkowa

`if` to fundament podjęcia decyzji w kodzie – jeśli warunek (wyrażenie logiczne) zwróci `true`, to wykona się określony blok kodu. W przeciwnym razie `else` możemy wykonać inny blok instrukcji. Załóżmy, że tworzymy system logowania do aplikacji przeznaczonej tylko dla osób pełnoletnich. W tym celu pobierzemy wiek użytkownika – jak to w PHP zmienna poprzedzona jest operatorem `$` – po czym sprawdzamy, czy spełnia wymagania:

```
$wiek = 17; // w praktyce pobrane z formularza albo bazy danych
if ($wiek >= 18) {
    echo "Witaj w systemie dla osób pełnoletnich.";
} else {
    echo "Dostęp zabroniony. Musisz mieć co najmniej 18 lat.";
}
```

Jeśli zmienna

`$wiek` spełnia warunek, czyli zawiera wartość równą co najmniej 18, to użytkownik zostanie wpuszczony do systemu.

Obsługa wielu scenariuszy – elseif albo else if

Możemy także, dzięki istnieniu konstrukcji

`elseif` rozbudować logikę skryptu, tak aby obsługiwał kilka scenariuszy:

```
$ocena = 4;
if ($ocena == 6) {
    echo "Celujący - gratulacje!";
} elseif ($ocena == 5) {
    echo "Bardzo dobry - świetny wynik!";
} elseif ($ocena == 4) {
    echo "Dobry - całkiem nieźle.";
} elseif ($ocena == 3) {
    echo "Dostateczny - zaliczone, ale mogło być lepiej.";
} else {
    echo "Konieczna poprawa!";
```

```
}
```

W języku PHP mamy możliwość zapisania takiej instrukcji zarówno jako

`elseif` (jednym słowem), jak również `else if` (dwoma słowami) – są to wersje zamienne, jednak tylko pod warunkiem, że korzystamy ze standardowych nawiasów klamrowych: `{ ... }` do oznaczania bloków kodu dla każdego scenariusza.

Alternatywna składnia if: dwukropek + endif

W PHP możemy też zastosować alternatywną składnię instrukcji warunkowej

`if` używając do otwarcia bloku kodu dwukropka `:`, zaś do jego zakończenia klauzuli `endif;` – możemy zdecydować się na taki alternatywny zapis z dwukropkiem zamiast klasycznych nawiasów klamrowych:

```
<?php
$liczba = 10;
if ($liczba > 0):
    echo "Liczba jest dodatnia";
elseif ($liczba < 0):
    echo "Liczba jest ujemna";
else:
    echo "Liczba jest równa zero";
endif;
?>
```

Kiedy stosujemy zapis alternatywny z dwukropkiem, to konieczne będzie zawsze zapisywanie

`elseif` jednym słowem – próba użycia `else if` jako dwóch wyrazów zakończy się błędem składniowym (*ang. parse error – błąd parsowania*).

Częsty błąd składniowy na egzaminach

Jednym z częstych błędów osób początkujących jest mylenie (przy zapisywaniu warunków logicznych) operatora przypisania

`=` z operatorem porównania `==`:

```
<?php
$login = $_POST['login']; // np. z formularza
$haslo = $_POST['haslo'];
if ($login = "admin" && haslo = "t4jn3#h45l0#admin") {
    echo "Zalogowano jako administrator!";
} else {
    echo "Niepoprawny login!";
}
?>
```

Powyższy kod zawsze zaloguje nas na konto *admin* – niezależnie od wprowadzonych w polach wartości loginu i hasła – zamiast **porównać** wartości zmiennych ze wzorcami, my **przypisaliśmy** do nich nowe wartości. Takie działanie totalnie zmieniło logikę programu!

Łączenie warunków – operatory logiczne

W PHP, podobnie jak w JavaScript, mamy dostęp do klasycznych operatorów logicznych:

- `||` albo alternatywnie `or` – operator alternatywy (“lub”).
- `&&` albo alternatywnie `and` – operator koniunkcji (“i”).
- `!` albo alternatywnie `not` – operator negacji (“nie”).

Działanie tych operatorów oraz ich hierarchię (

`&&` ważniejszy niż `||`) wyjaśniliśmy już w lekcji: Instrukcje warunkowe w JS, stąd nie będziemy teraz powtarzać tych wyjaśnień, a jedynie zaznaczmy, iż zapis spójników logicznych w PHP może być symboliczny: `||` , `&&` , `!` albo słowny: `or` , `and` , `not` .

Są to zasadniczo równoważne zapisy, choć też zawsze wersje symboliczne posiadają nieco wyższy priorytet (na przykład są ważniejsze od operatora przypisania) aniżeli ich odpowiedniki słowne. Jednak ma to znaczenie tylko dla wartości logicznych wstawianych do zmiennych (co czynimy bardzo rzadko, ale zobaczmy konkretny przykład):

```
$wynik = true && false;  // do zmiennej $wynik trafi wartość false
$wynik = true and false; // do zmiennej $wynik trafi wartość true
```

Dlaczego przypisano zmiennej różne wartości, w zależności od tego czy użyto operatora słownego czy symbolicznego? Stało się tak, ponieważ:

- Operator `&&` ma wyższy priorytet niż operator przypisania `=` , więc najpierw zostanie obliczona wartość warunku `true && false` , a dopiero potem przypisana do zmiennej `$wynik` – w efekcie trafia tam wartość `false` .
- Operator słowny `and` posiada niższy priorytet niż operator przypisania `=` , więc najpierw interpreter wykona `$wynik = true` , a dopiero potem wykona się fragment: `and false` (co nie wpłynie na wartość już przecież wpisaną do zmiennej).

Natomiast są to oczywiście niuanse – w budowaniu klasycznych porównań w instrukcjach warunkowych różnicy w działaniu pomiędzy operatorem symbolicznym a słownym nie zauważymy. Różnica dotyczy ważności względem operatora przypisania, czyli

`=` .

Skrócony zapis warunku – operator trójargumentowy

Oczywiście PHP również oferuje zapis skrócony

`if` , czyli tzw. operator warunkowy trójargumentowy – przydatny, gdy chcemy przypisać wartość do zmiennej, zależnie od wartości warunku:

```
<?php
$wiek = 20;
$komunikat = ($wiek >= 18) ? "Pełnoletni" : "Niepełnoletni";
echo $komunikat;
?>
```

Jeśli warunek jest spełniony, to zmienna przyjmie wartość zdefiniowaną po operatorze

`?`, a w przeciwnym wypadku wartość zdefiniowaną po dwukropku.

Przykład praktyczny – kalkulator rabatu

Typowe zadania egzaminacyjne wykorzystujące instrukcje warunkowe zazwyczaj polegają na podjęciu kilku prostych decyzji. Na przykład podczas tworzenia prostego sklepu internetowego, mamy za zadanie zrealizować logikę przyznawania zniżki w zależności od kwoty zamówienia (im wyższa kwota, tym więcej rabatu otrzymuje klient):

- Zamówienie 500 zł lub więcej – 20% rabatu.
- Między 200 a 499 zł – 10% rabatu.
- Poniżej 200 zł – brak rabatu.

Z wiedzą z tej lekcji, zrealizowanie takich kilku "scenariuszy" nie powinno nam sprawić większych problemów:

```php
<?php
$kwota = 300; // Kwota zamówienia wyjęta z bazy danych lub formularza
if ($kwota >= 500) {
    $rabat = 0.2; // 20% rabatu
} elseif ($kwota >= 200 && $kwota < 500) {
    $rabat = 0.1; // 10% rabatu
} else {
    $rabat = 0; // Brak rabatu
}
$cenaPoRabacie = $kwota - ($kwota * $rabat);
echo "Kwota po rabacie: " . $cenaPoRabacie . " zł";
?>
```

Podsumowanie

- Instrukcja `if` pozwala podejmować decyzje w kodzie – jest absolutnie kluczowa dla realizowania logiki programu.
- `if...else` obsłuży dwa przypadki (scenariusze) wykonania.
- `if...elseif...else` umożliwi obsługę wielu warunków.
- Operatory logiczne `||`, `&&`, `!` oraz ich wersje słowne `or`, `and`, `not` służą do budowy złożonych warunków. Dzięki nim nie trzeba zagnieżdżać ifów, tylko łączyć warunki spójnikami.
- Operatory zapisywane słownie są mniej ważne od przypisania `=`.
- Operator trójargumentowy `(warunek) ? wartość_true : wartość_false` umożliwia stworzenie skróconego zapis warunku.
- Błędy typu `=` zamiast `==` mogą zupełnie zmienić działanie programu.