Pierwszy kod PHP

Zaczynamy od stworzenia nowego pliku, który nazwiemy index.php. Na początek napiszmy kilka linijek w tradycyjnym HTMLu, np.:

```
<html>
	<head>
		<title>Pierwszy skrypt php</title>
	</head>
	<body>
		Przykładowy tekst na stronie.
	</body>
</html>
```

Powyższego kod wyświetli na ekranie „Przykładowy tekst na stronie”. Teraz zobaczmy na przykładzie, jak wykonać te sama operacje za pomocą języka php…

```
<html>
	<head>
		<title>Pierwszy skrypt php</title>
	</head>
	<body>
		<?php
			echo "Przykładowy tekst na stronie";
		?>
	</body>
</html>
```

Zobaczmy, co zmieniło się w stosunku do poprzedniej wersji. Po pierwsze pojawiły się znaczniki **<?php** oraz **?>**. To chyba najbardziej kluczowa wiedza o php – żeby kod został przetworzony przez interpreter, musi znajdować się właśnie między takimi oznaczeniami. Teraz zobaczmy, co zostało między nimi wpisane. Komenda **echo** po prostu wyświetla nam na ekranie to, co wpiszemy jej jako argument. W tym przypadku jest to ten sam tekst, co w czystym HTML, więc efekt będzie taki jak poprzednio.

PHP zmienne i stałe

Jak w każdym języku programowania, tak również w PHP spotkamy się ze stałymi i zmiennymi.

Zmienna w PHP to nic innego jak kontener przechowujący pewna wartość. Nieważne, czy jest to wartość logiczna, liczba całkowita, wymierna czy też tekst. Ciekawostką w języku PHP jest **brak konieczności deklaracji zmiennych**, co jest wymagane w innych językach, jak C. **Stała** jest podobnym kontenerem do zmiennej, z tym, że jak sama nazwa wskazuje, jej wartości nie można zmienić. Stałej przypisujemy wartość tylko przy definicji.

Jak wygląda inicjacja stałych i zmiennych?

```
<?php
	define("NR_TEL","666258147");
	$nr_tel = "792297792";
?>

<html>
	<head>
		<title>Zmienne i stale</title>
	</head>
	<body>

		<?php
			echo NR_TEL;
			echo $nr_tel;
		?>
	</body>
</html>
```

Patrząc na powyższy kod widzimy dwa przypisania na samym początku. Pierwsze z nich to definicja stałej o strukturze – **define(„NAZWA_STALEJ”, „WARTOSC_STALEJ”);**, gdzie NAZWA_STALEJ to dowolny ciąg liter, cyfr oraz znaku podkreślenia, a WARTOSC_STALEJ to nadana wartość. Dodatkowo nazwa nie może zaczynać się liczbą. Z racji tego, że PHP rozróżnia wielkość liter, stałe NR_TEL i NR_Tel to dwa różne kontenery. Linijka niżej to definicja zmiennej. Nazwy zmiennych kierują się tymi samymi zasadami, co stałych, z tą jedną różnicą, że nazwę zmiennej musimy poprzedzić znakiem **$**.

Do nadawania wartości zmiennym używa się operatora przypisania ” = „. Jego zadanie polega na przypisaniu wartości znajdującej się po prawej stronie operatora do zmiennej po lewej stronie.

Użycie stałych i zmiennych w PHP

Stworzyliśmy naszą zmienną oraz stałą, przypisaliśmy im wartości i co dalej z nimi począć? Oczywiście można je wyświetlić na ekranie za pomocą komendy **echo**, tak jak zostało to pokazane na przykładzie. Nie jest to jednak jedyna możliwość. Stała lub zmienna jest traktowana przez PHP jako liczba, ciąg znaków lub wartość logiczna – zależnie co jej przypisaliśmy. Skoro tak, to przypisując zmiennej **$liczba = 5;** będziemy mogli jej używać tak jak tradycyjnej liczby. Będzie można ją dodać, odjąć, mnożyć, dzielić itp.

Jeszcze jedna istotna sprawa odnośnie przypisywania wartości. Mianowicie możemy przypisać zmiennej **$zm = 15;** lub **$zm = „15”;.** Wszystkie ciągi znaków (potocznie zwane stringami) są zapisywane z wykorzystaniem znaków cudzysłowia („) lub apostrofu (‚). W PHP nie deklarujemy typów zmiennych, dlatego interpreter naszego kodu musi w jakiś sposób wywnioskować, czy chcemy traktować 15 jako liczbę, czy też jako stringa. Istnieje oczywiście coś takiego jak rzutowanie typu zmiennej, ale o tym, kiedy indziej.

Istnieje jeszcze grupa zmiennych formularza. Pisząc formularz w HTML należy później obsłużyć wypełnione pola, np. wysłać maila lub zalogować użytkownika. Istnieją dwie metody

wysyłania danych – **POST** oraz **GET**. Do obsłużenia danych wysyłanych pierwszą metodą służy zmienna **$_POST[‘nazwa_pola’]**. Odpowiednio dla drugiej metody będzie to **$_GET[‘nazwa_pola’].** Mając więc pole tekstowe o nazwie **„imie’** w formularzu **GET**, po wysyłaniu formularza pojawi się zmienna **$_GET[‘imie’]** zawierająca wpisaną treść przez użytkownika.

PHP Operatory – zestawienie

Pierwszą kategorią operatorów PHP, są **operatory arytmetyczne**. Ich użycie jest bardzo intuicyjne, a działanie niemal oczywiste.

Chodzi tutaj o dodawanie, odejmowanie, mnożenie i dzielenie liczb. Dodatkowo do tej grupy zalicza się operator zwracający resztę z dzielenia. Ich zestawienie wygląda następująco:

- ” + ” zwraca sumę dwóch liczb lub ciągów,
- ” – ” zwraca różnicę,
- ” * ” zwraca iloczyn,
- ” / ” zwraca iloraz,
- ” % ” zwraca resztę z dzielenia.

Drugą kategorię stanowią **operatory porównania**. PHP umożliwia nam sprawdzenie, czy dwie zmienne są sobie równe, lub jeśli nie, to która jest większa, a która mniejsza. Do tego celu służą właśnie operatory porównania. Są nimi:

- ” = = ” sprawdza, czy dwie zmienne są równe, co do wartości,
- ” != ” sprawdza, czy zmienne są różne co do wartości,
- ” = = = ” sprawdza, czy zmienne są identyczne,
- ” != = ” sprawdza, czy zmienne są nieidentyczne,
- ” > ” sprawdza, czy zmienna z lewej strony jest większa od zmiennej z prawej strony,
- ” < ” sprawdza, czy zmienna z prawej strony jest większa od zmiennej z lewej strony,

- ” >= ” sprawdza, czy zmienna z lewej strony jest większa bądź równa od zmiennej z prawej strony,
- ” <= ” sprawdza, czy zmienna z prawej strony jest większa bądź równa od zmiennej z lewej strony.

Trzecią są **operatory logiczne**. Służą głównie do sprawdzania warunków:

- ” ! ” operator zaprzeczenia (logiczne NOT),
- ” && ” operator koniunkcji (logiczne AND),
- ” || ” operator alternatywy (logiczne OR).

Odpowiednikiem operatorów arytmetycznych są tzw. **operatory inkrementacji i dekrementacji**, które w bardzo szybki i przyjemny sposób zwiększają lub zmniejszają wartość naszej zmiennej o 1. Dzielą się one na:

- ” $i++ ” **postinkrementację**, zwiększa wartość zmiennej o 1,
- ” ++$i ” **preinkrementację**, zwiększa wartość zmiennej o 1,
- ” $i– ” **postdekrementację**, zmniejsza wartość zmiennej o 1,
- ” –$i ” **predekrementację**, zmniejsza wartość zmiennej o 1.

Różnica między pre- i post- inkrementacją leży w momencie zwiększenia wartości. **Preinkrementacja** zwiększa wartość przed wykonaniem polecenia, natomiast **postinkrementacja** zwiększa wartość zmiennej po wykonaniu polecenia. Z dekrementacją jest analogicznie, z tym, że wartość jest zmniejszana.

Innym sposobem zmiany wartości zmiennej jest użycie **operatorów przypisania**. Chcąc zwiększyć daną wartość o pewną liczbę, zamiast pisać **$i = $i + 7**, możemy od razu przypisać zmiennej $i wartość o 7 większą, czyli **$i += 7**. Poniżej inne możliwości przypisania zmienionej wartości:

- ” $i+=5 ” zwiększenie wartości o 5,

- ” $i-=5 ” zmniejszenie wartości o 5,
- ” $i*=5 ” przypisanie wartości 5 razy większej,
- ” $i/=5 ” przypisanie wartości 5 razy mniejszej,
- ” $i%=5 ” przypisanie wartości reszty z dzielenia zmiennej przez 5.

Operator ciągu ” . „ łączy nam dwa ciągi w jeden. Np.

$x = „Kod „

$y = „PHP”,

echo $x.$y

wyświetli nam na ekranie „Kod PHP”.

Stosowanie komentarzy w PHP

Nie dotyczą tylko PHP, ale każdego języka programowania. Zasada jest prosta – komentujmy wszystko, co może stać się niejasne po pewnym czasie. Na przykład napisanie kodu pewnej witryny zajęło tysiąc linijek, w których użyto pięćdziesięciu zmiennych.

Zleceniodawca przez pierwszy rok był bardzo zadowolony, lecz później stwierdził, że zamiast wyświetlać na stronie głównej dziesięć najnowszych produktów, chciałby, żeby było ich dwadzieścia i to w dodatku losowych. Zobaczmy więc jakie problemy możemy napotkać na swojej drodze.

Jeśli nazwaliśmy nasze zmienne $i, $j, $k, $zmienna1 itp. to bardzo ciężko będzie nam znaleźć odpowiedni fragment kodu. Musimy analizować krok po kroku składnię, żeby wywnioskować, co się dzieje w danym miejscu.

Wtedy z pomocą przychodzą nam komentarze. Jeśli komentowaliśmy skrupulatnie naszą pracę z pewnością szybko odszukamy odpowiedni fragment. Podsumowując, jeśli **$ilosc** jest odpowiedzialna za ilość produktów w sklepie, to należy ją obkomentować „Przechowuje ilość

wszystkich produktów w sklepie”. Wtedy łatwo będzie nam ją odszukać i dokonać potrzebnych modyfikacji.

Generalnie, dobrze napisany kod nie wymaga komentowania każdej zmiennej.

PHP Komentarze w praktyce

Istnieją dwa sposoby umieszczania komentarzy w kodzie PHP. Pierwszy z nich stosujemy, gdy chcemy obkomentować kilka lub więcej linijek tekstu. Umieszczamy wtedy taki blok tekstowy między znakami ” **/*** ” oraz ” ***/** „. Wszystko zawarte pomiędzy tymi znacznikami zostanie zignorowane przez interpreter podczas generowania kodu html.

Co ważne, komentarze nie zostaną wysłane do przeglądarki użytkownika.

Drugim sposobem komentowania jest umieszczenie tekstu za dwoma ukośnikami ” **//** „. Jest to sposób szybszy i wygodniejszy od pierwszego z racji, że nie trzeba umieszczać znaków zamykających komentarz. Wszystko znajdujące się w jednym wierszu po tym oznaczeniu będzie traktowane jako komentarz. Obsługuje on jednak tylko jedną linijkę, więc jeżeli mamy długi tekst składający się z kilku wierszy, należy użyć pierwszego typu.

Po tej solidnej dawce nowego materiału należałoby go uporządkować.

Włączam edytor do pisania dokumentów tekstowych i utwórz dwa nowe pliki o nazwie **sklep.html** oraz **zamowienie.php**.

Naszym zadaniem będzie napisanie aplikacji, która liczy sumę zamówienia składanego w sklepie internetowym, liczy podatek VAT 23% od tej kwoty, a następnie prezentuje na ekranie cenę netto i brutto zamówionych przedmiotów.

Ceny netto wszystkich produktów będą przechowywane w formie stałych o nazwie szablonowej **NAZWA_PRODUKTU**. Zmienne **$ile_nazwa_produktu** będą przechowywać informacje odnośnie ilości sztuk danego produktu, zamówionych przez klienta. Wysokość podatku VAT również będzie przechowywana w stałej – **P_VAT**. Takie rozwiązanie jest bardzo wygodne w przypadku późniejszych zmian podatku. Jeżeli zaistnieje potrzeba zmiany wartości wystarczy to zrobić raz przy deklaracji stałej, a w całym dalszym kodzie wartość zostanie zmieniona. Zrobimy prosty interfejs, zawierający formularz **POST**. Na podstawie wpisanych w nim danych zostanie obliczone zamówienie. W celach instruktażowych kolejne etapy obliczeń są przypisywane do nowych zmiennych. Nie jest to rozwiązanie optymalne, ale na pewno bardziej przejrzyste.

Przegląd rozwiązania

Najpierw plik **sklep.html**. Jego kod wygląda tak:

```
<html>
   <head>
      <title>Sklep odzieżowy</title>
   </head>
   <body>
      <form action="zamowienie.php" method="post"> //obsługa formularza w pliku zamowienie.php
        Liczba zamawianych koszulek: <input type="text" name="koszulki" size=3 maxsize=3 />
        Liczba zamawianych spodni: <input type="text" name="spodnie" size=3 maxsize=3 />
        Liczba zamawianych czapek: <input type="text" name="czapki" size=3 maxsize=3 />
        <input type="submit" value="złóż zamówienie" />
   </body>
</html>
```

Listing pliku zamowienie.php:

```
<?php
   define("KOSZULKA", 14.99); // cena koszulki jako stała
   define("SPODNIE", 45.99); // cena spodni
   define("CZAPKA", 9.63); // cena czapki
   define("P_VAT", 0.23); // wysokość podatku VAT
```

```
    $ile_koszulki = $_POST['koszulki']; // przypisanie zmiennych formularza
    $ile_spodnie = $_POST['spodnie'];
    $ile_czapki = $_POST['czapki'];
    $kwota_koszulki_netto = $ile_koszulki*KOSZULKA; // wartość netto zamówionych koszulek
    $kwota_spodnie_netto = $ile_spodnie*SPODNIE; // wartość netto spodni
    $kwota_czapki_netto = $ile_czapki*CZAPKA; // wartość netto czapek
    $kwota_zamowienia_netto = $kwota_koszulki_netto + $kwota_spodnie_netto + $kwota_cz
apki_netto; // cena netto całego zamówienia
    $kwota_koszulki_brutto = $kwota_koszulki_netto + $kwota_koszulki_netto*P_VAT; // wart
ość brutto koszulek
    $kwota_spodnie_brutto = $kwota_spodnie_netto + $kwota_spodnie_netto*P_VAT; // wart
ość brutto spodni
    $kwota_czapki_brutto = $kwota_czapki_netto + $kwota_czapki_netto*P_VAT; // wartość b
rutto czapek
    $kwota_zamowienia_brutto = $kwota_koszulki_brutto + $kwota_spodnie_brutto + $kwota
_czapki_brutto; // cena zamówienia brutto
?>

<html>
    <head>
        <title>Obsługa zamówienia</title>
    </head>
    <body>
        <?php
            echo "Cena netto zamówionych koszulek: ".$kwota_koszulki_netto."<br/>";
            echo "Cena netto zamówionych spodni: ".$kwota_spodnie_netto."<br/>";
            echo "Cena netto zamówionych czapek: ".$kwota_czapki_netto."<br/>";
            echo "Wartość netto całego zamówienia: ".$kwota_zamowienia_netto."<br/>";
            echo "Cena brutto zamówionych koszulek: ".$kwota_koszulki_brutto."<br/>";
            echo "Cena brutto zamówionych spodni: ".$kwota_spodnie_brutto."<br/>";
            echo "Cena brutto zamówionych czapek: ".$kwota_czapki_brutto."<br/>";
            echo "Wartość brutto całego zamówienia: ".$kwota_zamowienia_brutto."<br/>";
        ?>
    </body>
</html>
```

Omówienie skryptu

Plik **sklep.html** jest odpowiedzialny za wyświetlenie formularza z możliwością wpisania liczby zamawianych przedmiotów. Po wpisaniu i zgłoszeniu formularza dane zostają przesłane do pliku **zamowienie.php**. Tutaj na wstępie definiujemy stałe z cenami przedmiotów oraz wysokością podatku VAT. Później tworzymy zmienne z pól formularza, przesłanych ze sklepu.

Kwoty netto obliczamy po prostu mnożąc ilość zamówionych przedmiotów przez cenę netto przechowywaną w stałej. Następnie w celu obliczenia całości sumujemy trzy kwoty netto. Z cenami brutto robi się podobnie, z tym, że należy dodać wartość podatku, czyli 0,23*kwota.

To kończy naszą część obliczeniową aplikacji, teraz czas to wszystko wyświetlić. Wykorzystujemy do tego funkcję **echo,** oprócz tekstu wpisanego przez nas dodatkowo jako argument podajemy zmienną. Jedną i drugą część tekstu musimy jednak połączyć operatorem ” . „„ żeby interpreter poprawnie zrozumiał nasze intencje. Działa to w ten sposób, że przeglądarka wyświetli najpierw to, co jest w cudzysłowiu, później wyświetli wartość zmiennej, a następnie to, co jest w kolejnym cudzysłowiu.

Ćwiczenie

- dodaj do naszego sklepu dodatkowy artykuł – buty, nadaj mu cenę netto i zmodyfikuj wyświetlanie, by pokazywane były cztery przedmioty,
- stwórz formularz HTML, w którym wstawisz trzy pola tekstowe, które po wypełnieniu wyślesz do pliku .php; wyświetl wpisane teksty w różnych kombinacjach używając funkcji echo i operatora ” . „„
- zmodyfikuj plik zamowienie.php, dodaj formularz, w którym będzie można podać kwotę, jaką się płaci za zamówione towary; kwota, wraz z wartością brutto w polu typu hidden, zostanie wysłana do kasa.php, w którym zostanie obliczona reszta, jaką należy wydać od zapłaconej kwoty; wskazówka: wykorzystaj funkcję echo w atrybucie value pola typu hidden, by przekazać wartość brutto zakupionych produktów.