Szybki kurs PHP

Dostępny dzięki firmie Netteria.NET – tworzenie stron www, projektowanie stron internetowych, tworzenie oprogramowania na zamówienie.

- Wstęp
- Pierwszy program
- Zmienne
- Operatory w PHP
- Typy danych
- Tablice
- Stałe
- Wyrażenia decyzyjne
- Pętle w PHP
- Własne funkcje
- Cookie

Wstęp

PHP (PHP Hypertext Preprocesor) jest językiem programowania przeznaczonym do tworzenia dynamicznych i interaktywnych serwisów WWW. Jego ogromną zaletą jest fakt, że w zasadzie jest on niezależny od platformy systemowej. Program działający pod kontrolą systemu linux czy też unix powinien również bez problemu zadziałać pod windows. Jest to niewątpliwie wielka zaleta ponieważ pisząc program nie musisz specjalnie zawracać sobie głowy na jakim systemie zostanie on uruchomiony.

Program PHP można poznać po tym, że ich rozszerzenie jest najczęściej *.php lub *.php3. Istnieje wprawdzie możliwość takiego skonfigurowania serwera na którym działają programy PHP, że rozszerzenie może być dowolnie określone, np. *.html, jednak chyba doskonała większość ma właśnie te dwa rozszerzenia. Dodam że *.php3 jest rozszerzeniem dla starszej wersji języka.

Czym właściwie jest program napisany w PHP? Program PHP jest programem działającym po stronie serwera. Tam też się wykonuje a wyniki w postaci gotowego kodu HTML są przesyłane do klienta, czyli do przeglądarki która żądała dostępu do strony. To kolejna zaleta PHP. Po pierwsze, że wynik Twojej pracy w postaci kody źródłowego nie jest ogólnie dostępny (przecież nie zawsze chcesz pokazać innym swój kod) a po drugie nie obciążasz klienta a co za tym idzie nie musisz brać pod uwagę parametrów technicznych sprzętu na którym pracuje.

Aby moc zacząć pisać swoje pierwsze programy potrzebujesz jakiegokolwiek edytora tekstu oraz miejsca na serwerze obsługującym PHP. Istnieje także możliwość zainstalowania serwera na lokalnym komputerze i pisania oraz testowania programów nawet bez dostępu do sieci. Jeśli jesteś zmuszony skorzystać z takiego rozwiązania to polecam Fox Server - Pakiet dla Windowsa instalujący PHP, serwer Apache, baze danych MySQL i phpMyAdmin, lub PHPTriad.

Pierwszy program

Jeżeli uporałeś się już z zorganizowaniem sobie stanowiska do pracy z PHP (zdobyłeś miejsce na serwerze obsługującym PHP lub zainstalowałeś go na lokalnym komputerze) to napiszmy pierwszy program – nieśmiertelne "Witaj świecie"

```
<?php
  print("Witaj świecie\n");
?>
```

Zapisz ten program w pliku z rozszerzeniem *.php i wpisz adres do niego w przeglądarce

W wyniku działania naszego programu w przeglądarce powinien pojawić się napis: " Witaj świecie ". Wszystko to dzięki funkcji PHP print, która umożliwia wyświetlenie danych.

Co widać jeszcze ciekawego w tym króciutkim programie ? Widzimy tu dwa znaczniki: " <? php " i " ?> ". Znaczniki te informują interpreter języka PHP na serwerze gdzie zaczyna się nasz program i gdzie kończy. I właśnie to co znajduje się pomiędzy tymi znacznikami wykonuje i przesyła do przeglądarki jako wynik w postaci kodu HTML. Kod PHP zazwyczaj łączy się z kodem HTML. Wygląda to tak:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
<TITLE> Mój pierwszy program</TITLE>
<META NAME="Author" CONTENT="Jan Kowalski ">
<META NAME="Keywords" CONTENT="php, skrypt, prpgram ">
<META NAME="Description" CONTENT="Strona z moim pierwszym
programem w PHP">
</HEAD>
<BODY>

<?php
  print("Witaj świecie\n");
?>
```

```
</BODY>
</HTML>
```

W ten sposób nasz program stanowi integralną część strony. Na serwerze zinterpretowane zostanie jako kod PHP to co jest między znacznikami " <?php " i " ?> ". Oczywiście do klienta zostanie przesłane wszystko co znajduje się w dokumencie czyli też kod HTML których użyliśmy, po prostu interpreter wyśle je razem z wygenerowanym przez siebie z kodu PHP kodem HTML. Pamiętaj jednak, że plik taki również musi być zapisany z rozszerzeniem *.php.

Zmienne

W PHP zmienne rozpoczynają się od znaku $. Przypisanie wartości zmiennej odbywa się za pomocą operatora przypisania którym jest znak =. Wygląda to w następujący sposób:

```
$miasto = "Londyn";
$liczba = 5;
```

Zmiennej $miasto nadaliśmy wartość Londyn a zmienna $liczba ma teraz wartość numeryczną 5. Pamiętaj o tym że zmienne numeryczne nie są objęte cudzysłowami.

W PHP nie istnieje ograniczenie co do długości nazwy zmiennych, wiadomo jednak że nie nikt nie będzie tworzył nazw o jakiejś niesamowicie duże ilości znaków.

Istnieją jednak inne ograniczenia co do nadawania nazw zmiennym. Nazwa zmiennej powinna się zaczynać od litery lub znaku podkreślenia, po których następuje litera, liczba lub znak podkreślenia.

Należy również pamiętać, że PHP rozróżnia wielkość liter, zmienna $numer i $Numer będą potraktowane jako dwie różne zmienne.

W PHP prócz zmiennych definiowanych przez użytkownika dostępne są również zmienne tworzone przez system. Jednak nie zawsze wszystkie są dostępne. Zależy to od serwera na którym pracujesz.

Zasięg zmiennych

W PHP zmienne globalne mają taki sam zasięg. Zasięg ten rozciąga się również na pliki dołączone. Zmienne globalne aby mogły być użyte wewnątrz funkcji muszą być zadeklarowane jako globalne:

Plik lib.inc.php

```
<?php
$zmienna = 123;
?>
```

Nasz plik z kodem. Powiedzmy index.php

```
<?php
include( "lib.inc.php ") //dołączamy plik w którym znajduje się
nasza zmienna

function wypisz_zmienna(){
global $zmienna; //deklaracja wewnątrz funkcji zmiennej jako
globalnej

echo $zmienna;
}
wypisz_zmienna();
?>
```

W wyniki wykonania tego programu powinno wyświetlić się nam 123.
W PHP istnieją również zmienne statyczne, które zadeklarowane wewnątrz funkcji umożliwiają utrzymanie swojej wartości pomiędzy kolejnymi wywołaniami tej funkcji.

```
<?php
function static(){
static $zmienna = 0; //deklaracja zmiennej jako statyznej
echo $zmienna;
$zmienna++;
}
static();
?>
```

Każde wywołanie naszej funkcji static zwiększy wartość zmiennej o jeden.

Operatory w PHP

W PHP dostępne są następujące operatory:

1. Operatory arytmetyczne
2. Operatory bitowe
3. Operatory porównania
4. Operatory zwiększania i zmniejszania
5. Operatory logiczne
6. Operatory przypisania
7. Operator trójskładnikowy
8. Operator wykonania
9. Operator kontroli błędów

Opis operatorów od punktu 1 do 6 znajduje się w poniższych tabelach:

Operatory arytmetyczne

Operator	Nazwa	Przykład	Wynik
+	Dodawanie	$a + $b	Suma $a i $b
-	Odejmowanie	$a - $b	Różnica $a i $b
*	Mnożenie	$a * $b	Iloczyn $a i $b
/	Dzielenie	$a / $b	Iloraz $a i $b
%	Reszta z dzielenia	$a % $b	Reszta z dzielenie $a przez $b

Operatory bitowe

Operator	Nazwa	Przykład	Wynik
&	Iloczyn bitowy	$a & $b	Bity ustawione w $a i $b są ustawione
|	Suma bitowa	$a | $b	Bity ustawione w $a lub $b są ustawione
^	Różnica symetryczna	$a ^ $b	Bity ustawione w $a lub $b, ale nie w obu na raz są ustawione
~	Negacja	~$a	Bity ustawione nie są teraz ustawione i odwrotnie
<<	Przesunięcie w lewo	$a << $b	Przesunięcie bitów w $a w lewo o $b kroków
>>	Przesunięcie w prawo	$a >> $b	Przesunięcie bitów w $a w prawo o $b kroków

Operatory porównania

Operator	Nazwa	Przykład	Wynik
==	Równy	$a == $b	True, jeżeli $a jest równe $b
===	Identyczny	$a === $b	True, jeżeli $a jest równe $b i są one tych samych typów
!=	Różny	$a != $b	True, jeżeli $a jest różne od $b
<	Mniejszy	$a < $b	True, jeżeli $a jest mniejsze od $b
>	Większy	$a > $b	True, jeżeli $a jest większe od $b

<=	Mniejszy lub równy	$a <= $b	True, jeżeli $a jest mniejsze lub równe $b
>=	Większy lub równy	$a >= $b	True, jeżeli $a jest większe lub równe $b

Operatory zwiększania i zmniejszania

Operator, przykład	Nazwa	Wynik
$a++	Postinkrementacja	Zwraca $a, a następnie zwiększa $a o jeden
++$a	Preinkrementacja	Zwiększa $a o jeden i zwraca $a
$a--	Postdekrementacja	Zwraca $a, a następnie zmniejsza $a o jeden
--$a	Predekrementacja	Zmniejsza $a o jeden i zwraca $a

Operatory logiczne

Operator	Nazwa	Przykład	Wynik
and	Iloczyn logiczny	$a and $b	True, jeżeli $a i $b mają wartość True
or	Suma logiczna	$a or $b	True, jeżeli $a lub $b mają wartość True
xor	Różnica symetryczna	$a xor $b	True, jeżeli $a lub $b mają wartość True, ale nie razem
!	Negacja	!$a	True, jeżeli $a nie jest True
&&	Iloczyn logiczny	$a && $b	True, jeżeli $a i $b mają wartość True
||	Suma logiczna	$a || $b	True, jeżeli $a lub $b mają wartość True

Operatory przypisania

Operator	Przykład	Wynik
=	$a = $b	Przypisuje wartość $b do $a.
+=	$a += $b	Przypisuje wartość ($a+$b) do $a. Jest to identyczne z $a=$a+$b.
-=	$a -= $b	Przypisuje wartość ($a-$b) do $a. Jest to identyczne z $a=$a-

		$b.
*=	$a *= $b	Przypisuje wartość ($a*$b) do $a. Jest to identyczne z $a=$a*$b.
/=	$a /= $b	Przypisuje wartość ($a/$b) do $a. Jest to identyczne z $a=$a/$b.
.=	$a .= $b	Przypisuje wartość ($a.$b) do $a. Jest to identyczne z $a=$a.$b.
%=	$a %= $b	Przypisuje wartość ($a%$b) do $a. Jest to identyczne z $a=$a%$b.
|=	$a |= $b	Przypisuje wartość ($a\|$b) do $a. Jest to identyczne z $a=$a|$b.
&=	$a &= $b	Przypisuje wartość ($a&$b) do $a. Jest to identyczne z $a=$a&$b.
^=	$a ^= $b	Przypisuje wartość ($a^$b) do $a. Jest to identyczne z $a=$a^$b.
<<=	$a <<= $b	Przypisuje wartość ($a<<$b) do $a. Jest to identyczne z $a=$a<<$b.
>>=	$a >>= $b	Przypisuje wartość ($a>>$b) do $a. Jest to identyczne z $a=$a>>$b.

Operator trójskładnikowy jest identyczny jak w C czy C++. Jego składnia jest następująca:

```
$zmienna = (wyrazenie_1) ? (wyrazenie_2) : (wyrazenie_3);
```

Zmienna będzie miała wartośćwyrażenie_2jeżeli wyrażeniewartość_1będzieTRUE(czyli prawdziwe) w przeciwnym wypadku zmienna będzie miała wartośćwyrażenie_3.

Operator wykonania oznaczamy znakiem ` (jest to znak umieszczony na tym samym klawiszu co znak tyldy) Wyrażenie otoczone tymi znakami jest wykonywane na serwerze a zwracana wartość przypisywana zmiennej. Przykładowo aby przypisać do zmiennej$listbieżącą zawartość katalogu możemy wykonać:$list = `ls -l`. W ten sposób nasza zmienna listing zawiera zawartości bieżącego katalogu na serwerze.

Operator kontroli błędów oznaczamy znakiem @. Umieszczenie tego operatora przed wyrażeniem powoduje że błędy nie będą wyświetlane w oknie przeglądarki a zostaną zapamiętane w zmiennej globalnej$php_errormsg. Należy jednak pamiętać, że zmienna ta jest nadpisywana przez kolejne błędy. Do prawidłowego użycia tego operatora konieczna jest uaktywniona na serwerze opcjatrack_errors.

Typy danych

PHP sam podczas interpretowania strony ustawia typ zmiennej. To nie programista jawnie ustala jej typ a jest on ustalany przez PHP w zależności od kontekstu w jakim owa zmienna została użyta.

W PHP rozróżnia się następujące typy danych:

- string (text)

- integer (liczby całkowite)
- double (liczby zmiennoprzecinkowe)
- array (tablice)
- object (obiekty)
- typ nieokreślony

String jest typem przechowującym dane tekstowe. Zarówno pojedyncze słowa jak i całe zdania. Wszystko to co jest zapisane w cudzysłowach, jest traktowane jako tekst (ciąg). Dotyczy to również danych numerycznych.

```
$imie = "Adam";
$wiek = "25";
```

Zarówno zmienna$imiejak i zmienna$wieksą typu string i przechowują ciąg. Jeżeli wykonasz operacji arytmetycznych na zmiennych przechowujących liczby w postaci ciągu (czyli zapisanych między cudzysłowem), to PHP dokona niejawnej konwersji na typ numeryczny.

```
$zmienna1 = "10";
$zmienna2 = "5";
$zmienna3 = $zmienna1 + $zmienna2;
```

Wartośćzmienna3wyniesie 15. PHP dokona konwersji do typu numerycznego i mimo, że nasze zmienne były ciągami operacja zostanie wykonana jak na liczbach. Z tego powodu aby móc łączyć ciągi musimy posłużyć się innym operatorem.

Operatorem wykorzystywanym do łączenia ze sobą ciągów jest operator . (kropka).

```
$imie = "Adam";
$wiek ="25";
$osoba = $imie . $wiek;
```

W ten sposób w zmiennej $osoba zostanie zapisany ciąg Adam25.

Ciąg może być również ujęty w apostrofy (’). Różnica polega na tym że w ciągu objętym cudzysłowem zmienne zapisane w ciągu zostają zamienione na ich wartości a w ciągu objętym apostrofami nie są zastępowane.

Aby zapisać znaki specjalne w ciągu zawartym w cudzysłowie musimy użyć znaku lewego ukośnika (\) Poniższa tabela przedstawia znaki specjalne w ciągu objętym cudzysłowami.

Sekwencja	Znaczenie
\n	nowa linia
\r	powrót karetki
\t	tabulacja
\\	lewy ukośnik
\"	cudzysłów
\$	znak dolara

W ciągu objętym apostrofami jedyne dopuszczalne sekwencje to ukośnik (\\) oraz apostrof (\’).

Numeryczne typy danych to integer i double. Typ integer reprezentuje liczby całkowite natomiast typ double liczby zmiennoprzecinkowe (do zapisu liczby zmiennoprzecinkowej używamy kropki po części całkowitej a nie przecinka).

```
$a = 5;
$b = 2.5;
$c = $a + $b;
```

Dodaliśmy do siebie zmienną typu całkowitego (integer) oraz zmiennoprzecinkowego (double). Wynik tej operacji został zapisany do zmiennej$ci wynosi 7.5.

Na danych typu numerycznego można wykonywać wszelkie operacje arytmetyczne. Ich kolejność wykonywania jest dokładnie taka sama jak w matematyce.

Kolejne typy danych wymagają nieco szerszego omówienia dlatego poświęcę im osobne rozdziały kursu.

Tablice

Jednym z typów danych w PHP są tablice. Tablicami nazywamy zbiór zmiennych o tej samej nazwie, jednak zmienne te mają inny indeks. Zmienne znajdujące się w tablicy nazywamy elementami tablicy.

```
$imiona[0] ="Adam";
$imiona[1] ="Ewa";
```

W ten sposób utworzyliśmy dwuelementową tablice imiona. Pod indeksem 1 znajduje się element o wartości Adam a pod indeksem 2 Ewa. Dokładnie tak samo inicjujemy tablice wartościami numerycznymi.

```
$numer[1] = 123;
```

W PHP nie jest wymagane podawanie typu danych w tablicy (tak jak nie jest wymagane podawanie typu zmiennych). Nie trzeba też podawać rozmiaru tworzonej tablicy. W tym miejscy PHP różni się od wielu innych języków gdzie tego typu informacje programista jest obowiązany jawnie podać.

Indeksy w tablicy wcale nie musza być po kolei. Można opuścić dowolny indeks.

```
$imiona[2] ="Adam";
$imiona[15]="Ewa";
```

Aby wyświetlić konkretny element musisz podać jego indeks.

```
echo $imiona[15];
```

W ten sposób zostanie wyświetlony ciąg Ewa.

Elementem tablicy może być również wartość innej zmiennej.

```
$imie = "Adam";
$imiona[1] = $imie;
```

Indeksy w tablicy nie koniecznie musza być numeryczne. Dopuszczalne też jest używanie ciągów znaków. Tablice tego typy nazywa się tablicami asocjacyjnymi.

```
$kraje["pl"] = "Polska";
$kraje["de"] = "Niemcy";
```

Aby odwołać się do elementów tablicy asocjacyjnej można podawać indeksy tablicy zarówno z cudzysłowach jak i bez nich

```
echo $kraje["pl"];
```

oraz

```
echo $kraje[pl];
```

w efekcie przyniosą ten sam skutek. W tym wypadku wyświetlenie ciągu Polska.

Do tej pory dokonywaliśmy inicjalizacji przy użyciu jawnie podanych indeksów. Można również inicjalizować tablice bez ich podawania

```
$imiona[] = "Adam";
$imiona[] = "Ewa";
```

Przy takiej inicjalizacji PHP samo zadba o indeksy. Będą to kolejne wartości numeryczne (liczone od 0). Należy bezwzględnie pamiętać o nawiasach kwadratowych, ponieważ ich opuszczenie nie pozwoli PHP odróżnić zmiennej tablicowej od zwykłej zmiennej. Skutkiem tego będzie nadpisanie wartości przez kolejną przez nas podaną.

Tablice wypełnić możemy również przy pomocy funkcji**array()**.

```
$imiona = array("Adam","Ewa","Ula","Magda");
```

W ten sposób mamy cztero elementową tablice imiona, której indeksy są liczone od 0 do 3. Jeżeli przeszkadza Ci to że indeksy są liczone od 0 możesz użyć operatora =>, który pozwala na określenie początkowego indeksu tablicy.

```
$imiona = array(1=> "Adam","Ewa","Ula","Magda");
```

W ten sposób indeksy w tablicy będą liczone od 1 do 4.

W przypadku tablic asocjacyjnych inicjalizowanych za pomocą funkcji**array()**użycie operatora => jest wręcz niezbędne.

```
$kraje array("pl" => "Polska", "de"=> "Niemcy", "sk" => "Słowacja");
```

Ten sposób inicjalizowania tablic asocjacyjnych jest szczególnie przydatny podczas tworzenia tablic o dużych rozmiarach.

Stałe

Czasem w programie potrzebujemy reprezentacji która nie ma podlegać modyfikacji. Przykładowo wiadomo że tuzin to 12. W takim przypadku potrzebujemy stałego obiektu (nie obiektu w rozumieniu programowania obiektowego) przechowującego tą informacje. W PHP możemy to osiągnąć poprzez stałe. Do definicji stałej niezbędne jest specjalne słowo kluczowedefine.

Oto jak powołać do życia stałą:

```
define("TUZIN",12);
```

I tak oto mamy stałąTUZINo wartości 12. Odwołanie się do tej stałej może wyglądać w ten sposób:

```
echo "Tuzin jest to " . TUZIN;
```

Wynikiem będzie wyświetlenie "Tuzin jest to 12". Przyjęto aby nazwy stałych pisać wielkimi literami. Istnieje jednak jedna zasadnicza różnica w użyciu zmiennych i stałych. Wynika to z faktu, że stałe nie są poprzedzone znakiem dolara. Konsekwencją tego faktu jest to że PHP nie może ich odróżnić od zwykłego tekstu w procesie zmiany nazwy stałej na wartość. Dlatego użycie stałej w cudzysłowie spowoduje wyświetlenie nazwy stałej a nie jej wartości. Dlatego nazwa stałej musi zawsze występować poza cudzysłowami. W przykładzie wyżej w tym celu użyłem operatora złączenia.

Wyrażenia decyzyjne

Często podczas programowania zachodzi potrzeba podjęcia decyzji jaka część kodu ma się wykonać w zależności od wystąpienia jakiegoś zdarzenia. PHP umożliwia podjęcie przez program takiej decyzji dzięki wyrażeniom sterującym przepływem programu.

Instrukcja if

Ogólna postać wyrażenia if jest następująca:

```
if(warunek) instrukcja.
```

Wyrażenie if pozwoli na wykonanie kodu tylko gdy warunek jest prawdziwy w przeciwnym przypadku instrukcja zostanie zignorowana. W instrukcji if można oczywiście umieścić więcej niż jedną instrukcje, należy jednak wtedy je zamknąć w blok kodu objęty nawiasami klamrowymi.

```
<?php
$a=1;
$b=10
if($a > 0){
   echo "a jest większe od zera i wynosi $a";
   echo "b wynosi $b";
}
?>
```

W naszym przykładzie warunek jest prawdziwy więc wykonają się zawarte w bloku dwie instrukcje. Gdybyśmy jako warunek wpisali na przykład$a < 0, to oczywiście nie wykonałoby się nic.

Podczas budowania warunków używać będziesz operatorów porównania oraz logicznych, które omówione zostały w poprzednich rozdziałach. Bardziej skomplikowany warunek może wyglądać na przykład tak:

```
if($a < 0 && $b !=5) echo "warunek jest prawdziwy";
```

Mamy tu dwa operatory porównania (< jest mniejsze i != -nie równa się) oraz jeden operator iloczynu logicznego (&& and). Warunek będzie prawdziwy jeżeli zmienna a będzie mniejsza od zera i b nie będzie równe 5.

Istnieją sytuacje gdy chcemy wykonać część kodu gdy warunek jest prawdziwy ale też chcemy wykonać jakąś część gdy jest fałszywy. W takim przypadku posłużyć się frazą else. Aby to zobrazować w prosty sposób zmodyfikujemy nasz ostatni przykład.

```
if($a < 0 && $b !=5){
      echo "warunek jest prawdziwy";
{
else{
      echo "warunek jest fałszywy";
}
```

Teraz gdy warunek nie będzie prawdziwy zostanie wyświetlony napis: "warunek jest fałszywy".

Ale to nie wszystko. Możemy też skonstruować takie wyrażenie, które umożliwi nam podejmowanie decyzji dla więcej niż jednego warunku.Wystarczy posłużyć się frazą elseif. Po raz kolejny zmodyfikujemy nasz przykład aby to wyjaśnić.

```
if($a < 0 && $b !=5){
   echo "warunek pierwszy jest prawdziwy";
{
elseif($a > 0 && $b==5){
   echo "warunek drugi jest prawdziwy";
}
else{
```

```
   echo "obydwa warunki są fałszywe";
}
```

Instrukcje if można również zagnieżdżać.

```
if($a == 1){
  echo "a jest równe jeden";
   if($b == 1){
   echo "b jest również równe jeden";
   }
   else{
   echo "b nie jest równe jeden;
   }
}
```

Instrukcja switch

Instrukcja switch umożliwia wykonywanie części kodu w zależności od wartości zmiennej podanej jako argument. Najlepiej przeanalizujmy sobie to na prostym przykładzie.

```
switch($wiek){
   case $wiek > 65:
   echo "Witaj emerycie";
   break;
   case $wiek > 50:
   echo "Niedługo zasłużony odpoczynek";
   break;
   case $wiek > 40:
   echo "Wspaniały wiek produkcyjny";
   break;
   default:
   echo "Oszczędzaj się. Jeszcze sporo napracujesz";
}
```

W zależności od wartości zmiennej $wiek wykona się odpowiednia część kodu. Jeżeli żadna wartość zmiennej nie będzie prawdziwa wykona się default (domyślna). Należy zwrócić uwagę na instrukcje break, która powoduje bezwarunkowe opuszczenie instrukcji. Gdybyśmy jej nie użyli wykonały by się również kolejne instrukcje występujące po tej dla której był prawdziwy warunek.

Pętle w PHP

Niejednokrotnie w naszych programach chcemy aby jakaś cześć kodu wykonywała się dopóki określony przez nas warunek jest prawdziwy. W tym celu PHP posiada instrukcje pętli.

Pętla while

Ogólna postać pętli while jest następująca:

```
while(warunek){
      blok instrukcji
}
```

Podczas działania pętli sprawdzany jest warunek. Jeżeli warunek ten jest TRUE (prawdziwy), to wykonywany jest blok instrukcji. Pętla działa tak długo jak długo warunek jest prawdziwy. Jeżeli na wstępie warunek jest fałszywy, to blok instrukcji nie wykona się ani razu.

```
while($a < 10){
      echo "Warość jest mniejsza od 10";
      $a++;
}
```

W powyższym przykładzie będzie wyświetlany napis: "Wartość jest mniejsza od 10" do momentu kiedy zmienna a osiągnie 10. W każdym przebiegu pętli zmienna jest zwiększana o jeden dzięki instrukcji$a++.

Pętla do while

Pętla do while jest bardzo podobna w działaniu do pętli while. Różni się ona jednak tym że warunek wykonywania sprawdzany jest na końcu.

```
do{
      blok instrykcji
}while(wyrażenie);
```

Nawet jeżeli na wstępie warunek nie będzie prawdziwy, to i tak raz wykona się blok instrukcji. Zakończenie wykonywania pętli oczywiście wystąpi w momencie kiedy warunek przestanie być prawdziwy.

Pętla for

Jest to niezwykle przydatna pętla w sytuacji kiedy musimy wykonać jakąś część kodu określoną ilość razy. Ogólna postać jest następująca:

```
for(wartość początkowa licznika; sprawdzanie wartości licznika; zwiększenie
albo zmniejszenie wartości licznika){
instrukcje
}
```

W praktyce wygląda to tak:

```
for($i=0;$i<10;$i++){
  echo $i;
}
```

W ten sposób wyświetlimy kolejne wartości zmiennej $i (od 0 do 9). Pętla for działa według następującego algorytmu:

- obliczana jest wartość początkowa licznika
- sprawdzana jest wartość licznika (wartość warunku)
- wykonywane są instrukcje zawarte w bloku
- obliczana jest kolejna wartość licznika (zwiększana lub zmniejszana).

Pętle for możemy na przykład wykorzystać do sekwencyjnego przeglądania tablicy.

```
<?php

  $imiona = array(1=> "Adam","Ewa","Ula","Magda");

  for($index=1; $index<= 4; $index++){
   echo "<br>$imiona[$index]";
  }

?>
```

Powyższy kod spowoduje wypisanie wszystkich imion z utworzonej przez nas tablicy imiona.

Pętla foreach

Pętle tą stosujemy, gdy mamy tablice o nieznanej ilości elementów pętla przebiega po wszystkich elementach tablicy. Pętla foreach posiada dwa formaty.

```
foreach($NazwaTablicy As $ElementTablicy){
  instrukcjie
}


foreach($NazwaTablicy As $IndexTablicy => $ElementTabllicy){
  instrukcje
}
```

Pierwsza postać pętli przebiega po podanej tablicy i w każdym przebiegu wartość bieżącego elementu jest przypisywana zmiennej$ElementTablicya wskaźnik bieżącego elementu tablicy jest przesuwany.

Druga postać różni się tylko tym, że do zmiennej$IndexTablicyjest przypisywany bieżący indeks elementu.

Spójrz na przykład zastosowania drugiego rodzaju pętli foreach.

```
<?php

$kraje =array("pl" => "Polska", "de"=> "Niemcy" , "uk"=> "Wielka Brytania");

  foreach($kraje As $domena => $wartosc){
   echo "$wartosc: domena \"$domena\" <br>";
}
?>
```

W wyniku tego programu zostanie wyświetlona lista krajów i odpowiadających im domen. Lista ta oczywiście będzie niezwykle skromna tak jak nasza tablica.

Własne funkcje

Funkcje definiowane przez użytkownika są niezwykle ważnym elementem optymalizacji kodu. Pisząc swoje skrypty w pewnym momencie na pewno zauważysz, że pewne jego części są powtarzane. Możesz to właśnie wyeliminować poprzez zdefiniowanie funkcji która będzie realizowała to zadanie.

W PHP funkcje muszą być zadeklarowane przed ich użyciem. Funkcje mogą przyjmować argumenty (domyślnie argumenty są przekazywane przez wartość) lub mogą nie przyjmować żadnych argumentów. Funkcja również może zwracać jakąś wartość lub nie.

```
<?php
function suma($a, $b){
  $c = $a + $b;
  return $c;
}
echo suma(5,5);
?>
```

Nasza funkcja suma jak widać przyjmuje dwa argumenty. W ciele funkcji sumuje wartości tych argumentów i wynik zwraca dzięki instrukcji return. Następnie wywołujemy to funkcje i podajemy jako argument funkcji echo. Powinniśmy zobaczyć w oknie przeglądarki wynik.

W ciele funkcji możemy również wywoływać inne funkcje.

```
<?php
function napis(){
  echo "Dziś mamy ".gmdate("d m Y");
}

napis();
?>
```

Mamy tu przykład wywołania funkcji w ciele funkcji przez nas definiowanej. Jednocześnie jest to funkcja która nie przyjmuje żadnych argumentów i żadnych wartości nie zwraca. Jej zadaniem jest wyświetlenie napisu informującego jaki mamy dzień.

Jeszcze mała uwaga na temat organizacji kodu. Definicje swoich funkcji najlepiej umieszczać w innym pliku, np. lib.inc.php a następnie dołączać go do plików w jakich zamierzamy używać zdefiniowanych przez nas funkcji poprzez wyrażenie include.

plik lib.inc.php:

```
<?php
function napis(){
  echo "Dziś mamy ".gmdate("d m Y");
}
?>
```

plik index.php:

```
<?
include("lib.inc.php"); //Wyrażenie dołącza plik. Jeśli nie znajduje się on
w tym samym katalogu pamiętaj aby podać jako argument całą ścieżkę do pliku.

napis();
?>
```

Cookie

Na pewno niejednokrotnie słyszałeś o cookie. Wiele się o nich mówi ale często są to opinie bzdurne i wyssane z palce przez userów internetu którzy nic o nich nie wiedzą. Tak naprawdę cookie to niewinne małe pliki tekstowe, które pozwalają na przechowanie niewielkiej ilości danych na komputerze klienta. Niestety nie jest to zbyt pewna metoda przechowywania danych. Powodem tego jest nietrwałość plików cookie (na przykład zmiana przeglądarki powoduje ich utratę) oraz nie wszyscy użytkownicy będą mieli włączoną ich obsługę. Z tego powodu jeżeli dane które chciałbyś przechować są krytyczne dla aplikacji nie powinieneś używać cookie (alternatywne metody omówione zostaną w dalszej części kursu).

Do czego mogą być potrzebne pliki cookie ? Można je wykorzystać np. do zapamiętania preferowanego przez użytkownika wyglądu witryny. Wykorzystywane są również w koszykach sklepowych i mechanizmach logowania w witrynie. Można również zapisać w nich ostatnią wizytę użytkownika na stronie.

PHP posiada tylko jedną funkcję przeznaczoną do tworzenia cookie,setcookie(). Należy bezwzględnie przestrzegać zasady, że funkcje setcookie() należy wywołać przed wysłaniem kodu HTML. Cookie są ustawiana przez nagłówki HTTP, które są wysyłane przed kodem HTML. Można również dokonać buforowania wyjścia w celu opóźnienia wysyłania danych do przeglądarki aż do chwili zdefiniowania wszystkich cookie.

Definicja funkcji wygląda następująco:

int setcookie(string nazwa, string wartość, int czas, string ścieżka, string domena, int bezpieczny)

Argument nazwa musi wystąpić bezwzględnie, wszystkie inne są opcjonalne. Wartości będące ciągiem można opuścić podając pusty ciąg (" ") natomiast wartości numeryczne podając zero. Tak naprawde najważniejsze są trzy pierwsze argumenty. Ciąg będący nazwą zmiennej druga wartość będąca wartością zmiennej natomiast trzecia wartość wskazuje wartość czasu po którym wygaśnie ważność cookie. Trzeci argument jest znacznikiem czasu Uniksa, który można uzyskać jako wynik funkcjimktime()lubtime(). Kolejne trzy argumenty określają: ścieżkę do plików dla jakich jest ustawione cookie, domene dla której jest ustawione cookie oraz liczba całkowita która jeśli ma wartość jeden powoduje że cookie będzie tylko przesyłane w przypadku transmisji za pomocą protokołu SSL.

Odczyt cookie jest niezmiernie prosty. PHP automatycznie umieszcza dane pochodzące z cookie w domenie globalnej i wartośći są dostępne jak zmienne globalne.

```
setcookie("zmienna","10", time() + 360);
```

W ten sposób zapisaliśmy do pliku cookie zmienną o wartości 10. Jeżeli zechcemy usunąć cookie możemy zrobić to na dwa sposoby:

wywołując funkcje setcookie() z nazwą cookie jakie chcemy usunąć

```
setcookie("zmienna");
```

lub podając czas wygaśnięcia cookie z przeszłości

```
setcookie("zmienna", "10", time() - 999);
```

Poprzednie wersje PHP wymagały w przypadku ustawiania wielu cookie za pomocą jednego skryptu wywołania setcookie() w odwrotnej kolejności do tej w jakiej miały być obsługiwane ciasteczka. W PHP4 nie ma już tej niedogodności. Należy wywoływać funkcje setcookie w tej samej kolejności w jakiej będą przetwarzane przez skrypt cookie.