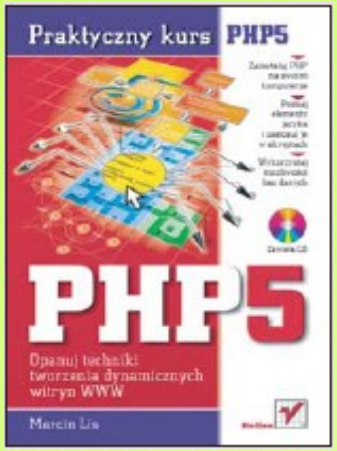
TECHNII PROGRAMOWANIA

Wykład 4:

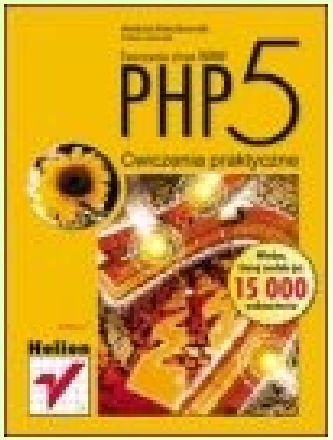
Języki programowania – PHP

Ryszard Myhan

PHP - bibliografia



PHP5. Praktyczny kurs
Marcin Lis
ISBN: 83-246-0307-7



PHP5. Tworzenie stron WWW.
Ćwiczenia praktyczne
Andrzej Kierzkowski,
Andrzej Kierzkowski
ISBN: 83-7361-361-7

PHP5. Programowanie
Rasmus Lerdorf, Kevin Tatroe,
Peter MacIntyre
ISBN: 978-83-246-0613-9

Czym jest PHP ?

 **PHP** jest jednym z wielu języków programowania, które umożliwiaj szybkie tworzenie dynamicznych stron internetowych.

 **PHP** to obiektowy, skryptowy język programowania zaprojektowany do generowania stron internetowych w czasie rzeczywistym.

 Oficjalna nazwa PHP to " **PHP: Hypertext Preprocessor** " a sam skrót pochodzi od *Personal Home Page.*

Czym jest PHP ?

PHP jest najczęściej stosowany do uruchamiania skryptów po stronie serwera WWW, ale może być również używany do przetwarzania danych z poziomu wiersza poleceń, a nawet do pisania programów pracujących w trybie graficznym.

W **PHP** kod nie jest przesyłany do przeglądarki klienta ale wykonywany przez serwer WWW (np. Apache, Xitami, IIS) a wyniki działania skryptu są przesyłane do przeglądarki już w kodzie języka HTML.

Historia PHP

- Pierwsza wersja PHP, rozpowszechniana pod nazwą PHP/FI (*Personal Home Page/Forms Interpreter*), została stworzona przez duńskiego programistę Rasmusa Lerdorfa w roku 1994.

- W 1997 roku projektem zainteresowali się dwaj izraelscy programiści, Zeev Suraski i Andi Gutmans. Zdecydowali oni, że przepiszą kod PHP od nowa, korzystając z pomocy istniejącej już wówczas społeczności PHP. W czerwcu 1998r ogłosili PHP 3.0

- W połowie 1999 roku został stworzony nowy silnik języka – Zend Engine, na bazie którego rozpoczęto pracę nad PHP 4. Ostatecznie PHP 4 zostało wydane w maju 2000 roku.

Historia PHP

- W 2002 roku Suraski i Gutmans ponownie rozpoczęli modernizację silnika Zend, mającą na celu dodanie do PHP modelu obiektowego, podobnego do tych, który istnieje w innych językach obiektowych.
- W lutym 2003 ukazała się wersja PHP 5.0.0 (stabilna wersja została wydana w lipcu 2004 roku) - pojawił się w niej całkowicie nowy model programowania obiektowego, co spowodowało utratę pełnej kompatybilności z poprzednimi wersjami PHP.
- W połowie roku 2005 rozpoczęto wstępne prace nad PHP 6 - obecnie ta wersja znajduje się w fazie projektowania, dostępne są jednak publiczne obrazy PHP w wersji źródłowej, jak również w wersji binarnej dla systemów Windows.

Dlaczego PHP ?

- **PHP** jest narzędziem niezależnym od systemu operacyjnego serwera WWW.
- **PHP** działa z większością webserwerów (Apache, Microsoft IIS, AOL Server, Netscape Enterprise Server, Xitami) dostępnych na różnych systemach operacyjnych (systemy unixowe, Windows itp.)
- **PHP** zapożycza najlepsze cechy i możliwości funkcjonalne języka C, Javy, Perl-a (składnia języka PHP jest bardzo podobna do składni języka C).
- **PHP** w odróżnieniu od analogicznych rozwiązań komercyjnych jest produktem darmowym udostępnianym na zasadach Open Source.

Możliwości funkcjonalne PHP

☑ Możliwości PHP nie ograniczają się do generowania danych wyjściowych w postaci kodu HTML.

☑ PHP umożliwia generację danych binarnych, w tym także obrazków JPEG, PNG oraz GIF.

☑ PHP pozwala na generowanie dokumentów PDF.

☑ PHP posiada narzędzia potwierdzania tożsamości, dostępne w protokole HTTP.

☑ W PHP istnieje możliwość przesyłania na serwer plików binarnych i tekstowych.

☑ PHP daje możliwość tworzenia i wykorzystywania cookies.

Możliwości funkcjonalne PHP

☑ Dysponuje możliwością obsługi protokołu SNMP, pozwalającego na monitorowanie wielu urządzeń: ruterów, koncentratory i serwery.

☑ PHP pozwala na korzystania z wielu standardów, takich jak HTML, LDAP, SMTP, SNMP, POP, IMAP.

☑ PHP zapewnia kompresję i dekompresję danych oraz posiada funkcje kryptograficzne.

☑ PHP umożliwia korzystanie z appletów (lub servletów) Java, obsługuje standard XML.

☑ PHP oferuje wsparcie dla wielu baz danych : Adabas, Dbase, Dbm, FilePro, Hyperwave, Informix, InterBase, mSQL, Microsoft SQL Server, MySQL Sybase, Oracle, PostgreSQL, Solid oraz ODBC

Co jest niezbędne do wykonania skryptu PHP?

☑ **Przeglądarka internetowa,**

☑ **Zainstalowany jeden z serwerów WWW, np. Apache, Xitami, IIS,**

☑ **Zainstalowany interpreter języka PHP (ang. parser)**

Uwaga:

Najlepiej zainstalować cały pakiet np. **KRASNAL ver. 2.7** lub **XAMPP Lite** 1.7.2 - posiadają wszystko, co potrzeba do uruchomienia, przetestowania skryptów.

Możliwości funkcjonalne PHP

- Skrypty są tworzone w językach programowania, a ich kod jest przekazywany przed wykonaniem do specjalnego oprogramowania – **interpretera** (parsera), który tłumaczy go na postać zrozumiałą w danym systemie operacyjnym, po czym skrypt ten jest wykonywany.
- Pliki PHP powinny mieć domyślne rozszerzenie **.php** lub **.html**.
- Skrypty PHP mogą być umieszczane w odrębnych plikach, mogą też być zawarte w treści kodu strony w języku HTML, podobnie jak kod Java Script.
- Interpretacją kodu skryptu PHP zajmuje się serwer WWW z wbudowanym interpreterem zwracając do przeglądarki wynik w postaci kodu HTML.

Umieszczanie kodu PHP w HTML

1. Metoda 1 (krótkie znaczniki)

```
<? echo "Skrypt PHP, najprostszy sposób”; ?>
```

2. Metoda 2 (preferowana)

```
<?php
echo "Prawie to samo, ale częściej stosowane”;
?>
```

3. Metoda 3 (znacznik skryptów)

```
<script type="text/php”>
echo "Metoda dla tych, co lubią dużo pisać”;
</script>
```

4. Metoda 4 (znaczniki ASP)

```
<% echo
"Metoda ASP, nie każdy serwer ją obsługuje”;
%>
```

PHP – pierwszy skrypt

otwarcie bloku kodu PHP

niby-funkcja zwracająca do przeglądarki podany w cudzysłowie tekst

zamknięcie bloku kodu PHP

```
<?php echo „To jest pierwszy skrypt"; ?>
```

funkcja, która powoduje wyświetlenie argumentu w oknie przeglądarki

średnik kończy każdą linię w skrypcie php inaczej interpreter zwróci błąd

- Procedury echo można użyć na dwa sposoby: echo ("tekst”); oraz echo "tekst”; - różnią się one tylko sposobem wyświetlania tekstu.
- W wypadku echo („tekst”); treści możemy wstawić, oddzielając je przecinkiem, np. echo („tekst1”, „tekst2”, „tekst3”);,
- a w echo „tekst”;, łącząc je kropkami, np. echo „tekst1”.”tekst2”.”tekst3”;.

Umieszczanie kodu PHP w HTML - przykład

```
<HTML>
        <HEAD>
                <TITLE>Przyklad 1</TITLE>
        </HEAD>
        <BODY>
                Aktualna data to:
                <?PHP
                   //wyświetl aktualną datę
                   print(Date("d-m-Y"));
                ?>
        </BODY>
</HTML>
```

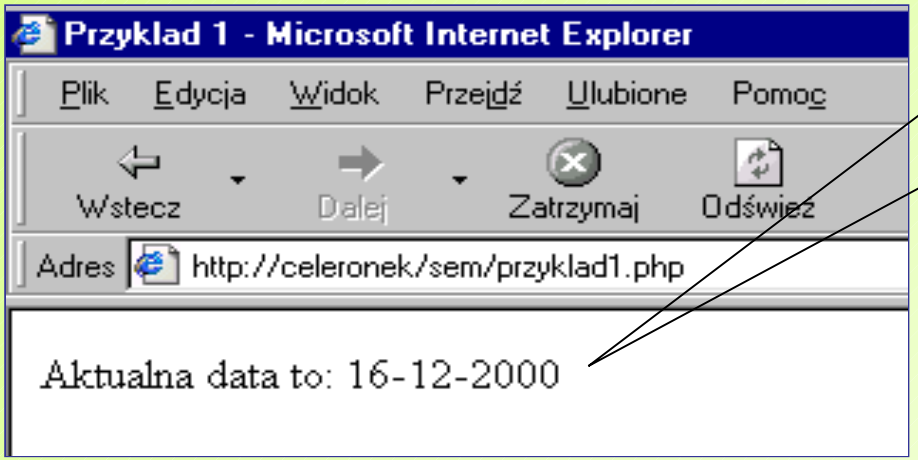
kod PHP

Umieszczanie kodu PHP w HTML - przykład

```
<HTML>
<HEAD>
<TITLE>Przyklad 1</TITLE>
</HEAD>
<BODY>
Aktualna data to:
16-12-2000</BODY>
</HTML>
```

Kod który dostaje przeglądarka nie zawiera PHP. Jest to czysty HTML, dzięki czemu nikt nie wie jak wygląda nasz skrypt PHP

Przeglądarka wyświetla zwykły kod HTM



PHP – identyfikatory

- Identyfikator to ciągi znaków służące do definiowania nazw zmiennych, stałych, funkcji, itp.
- Identyfikator musi spełniać nastepujące zasady:
  - Składać się z liter, cyfr, znaku podkreślenia, znaku dolara ($) ale nie może zaczynać się cyfrą lub zawierać polskich znaków diakrytycznych.
  - PHP rozróżnia wielkość liter, więc np. zmienna **$imie** nie oznacza tego co zmienna **$Imie**.
  - Nie wolno stosować jako identyfikatory nazw zastrzeżonych (słów kluczowych), np. nazw funkcji standardowych – **printf/echo**, **define**, itp.

PHP - komentarze

- Komentarze to specjalnie oznaczone fragmenty kodu, które są omijane przez parser.
- Służą do różnych celów – można w nich umieszczać np. uwagi do danego fragmentu skryptu, znaki copyright czy uwagi do innych członków grupy roboczej.
- Istnieją trzy sposoby oznaczania komentarzy:
  - // komentarz jednoliniowy, tylko do końca linii,
  - # komentarz jednoliniowy, sposób znany z shella,
  - /* komentarz wieloliniowy – można oznaczyć nim np. całe fragmenty kodu, aby tymczasowo wyłączyć je z wykonywania */

PHP - stałe

- Stałe przechowują dane, które nie będą modyfikowane podczas pracy programu.
- Definiuje się je za pomocą funkcji **define()**.

```
<?
define (”POLE”,200); // nazwa: POLE, wartosc: 200
echo (”Pole wynosi: ”.POLE.”cm”);
?>
```

PHP – stałe predefiniowalne

- PHP tworzy automatycznie kilka stałych gotowych do wykorzystania w skryptach:

__FILE__	zawiera nazwę pliku, który jest akurat przetwarzany.
__LINE__	zawiera numer akurat przetwarzanej linii skryptu.
PHP_VERSION	zawiera wersję akurat używanego parsera PHP,
PHP_OS	zawiera nazwę systemu operacyjnego, na którym uruchamiany jest parser PHP.

PHP – zmienne

- Zmienne są jednym z najważniejszych elementów języka programowania.
- Umożliwiają przechowywanie danych w pamięci komputera i wykonywanie na nich różnego rodzaju operacji.
- Zmiennych nie trzeba deklarować, gdyż typ zmiennej określany jest w czasie wykonywania programu i zależy od kontekstu, w jakim zmienna została użyta.
- Nazwa zmiennej musi być poprzedzona znakiem **$**:

```
$a = 2;          //zmienna typu całkowitego
$b = 1.2;        //zmienna typu rzeczywistego
$c = "tekst";    //zmienna typu tekstowego
```

PHP – zmienne środowiskowe

- Zmienne środowiskowe są parami nazwa-wartość istniejącymi w danej sesji użytkownika.
- W każdym skrypcie można użyć kilku zdefiniowanych i gotowych do użycia zmiennych - ich wartości zależą od ustawień serwera.
- Najważniejsze z nich to:
  - DOCUMENT_ROOT – zwraca katalog główny serwera WWW, gdzie umieszczony jest skrypt,
  - HTTP_ACCEPT_CHARSET – zwraca zawartość nagłówka „Accept-Charset” z aktualnego zapytania, jeśli taki istnieje, np. iso-8859-1,
  - HTTP_HOST – zwraca zawartość nagłówka „Host” z aktualnego zapytania, jeśli taki istnieje,

PHP – zmienne środowiskowe

- HTTP_REFERER – zwraca adres strony (jeśli taka istniała), która przekierowała przeglądarkę do naszej witryny.
- HTTP_USER_AGENT – zwraca zawartość nagłówka „User-Agent” z zapytania, jeśli taki istnieje - jest to ciąg informujący o przeglądarce, która została użyta do wyświetlenia bieżącej strony.
- REMOTE_ADDR – zwraca adres IP użytkownika, który wyświetlił bieżącą stronę,
- SCRIPT_FILENAME – zwraca ścieżkę do wykonywanego skryptu,

PHP – zmienne środowiskowe

- SERVER_ADMIN – zwraca adres administratora serwera WWW.
- SERVER_PORT – zwraca port na serwerze, którego użyto do połączenia. Dla standardowych połączeń będzie to wartość 80.
- SERVER_SIGNATURE – zwraca ciąg zawierający wersję i nazwę wirtualnego hosta, który jest dodawany do stron generowanych przez serwer.
- SCRIPT_NAME – zwraca ścieżkę do wykonywanego pliku - jest to przydatne w wypadku skryptów odwołujących się do samych siebie.

PHP – typy zmiennych

- W PHP istnieje pięć podstawowych typów zmiennych:
  - liczby całkowite (**integer**),
  - liczby rzeczywiste (**double**),
  - ciągi (**string**),
  - tablice (**array**),
  - obiekty (**object**).
- PHP, kiedy zachodzi taka potrzeba, automatycznie zamienia typ zmiennych w wyniku wykonywanych na nich operacji.
- Można też wymusić zmianę typu wykorzystując w tym celu procedurę rzutowania **cast** lub **settype()**.

PHP – zmiana typu zmiennej

- Procedura **cast** służy do jednorazowej zamiany typu zmiennej.

```
<?
        $liczbaRzeczywista = (real) $liczbaCalkowita;
?>
```

- W tym wypadku dozwolone typy rzutowań to:
  - (int), (integer) – rzutuj do typu całkowitego,
  - (real), (double), (float) – rzutuj do typu rzeczywistego,
  - (string) – rzutuj do ciągu,
  - (array) – rzutuj do tablicy,
  - (object) – rzutuj do obiektu.

PHP – zmiana typu zmiennej

- Druga możliwość to użycie funkcji **settype()**, która pobiera dwa argumenty.
  - pierwszy to nazwa zmiennej, której typ chcemy ustalić,
  - drugi to docelowy typ zmiennej. (dozwolone to „integer”, „double”, „string”, „array” i „object”).

```
<?
        $liczba = 10.3;
        settype($liczba, "integer");
        echo $liczba;
        // po wykonaniu zmienna przyjmie
        // wartość całkowitą10.
?>
```

Operator przypisania

Operatory to specjalne znaki, pozwalające na wykonywanie operacji na zmiennych.

➥ **Operator przypisania** służy przypisaniu wartości wyrażenia z jego prawej strony (wyrażenie, zmienna lub funkcja) do zmiennej z jego lewej strony.

```
<?
        $a = 5;
        // przypisuje liczbę 5 do zmiennej $a
        $A=$b4=$c=$d=4;
        /* w jednej linii można umieścić
        kilka przypisań. */
?>
```

Inne sposoby przypisania

- Operatory te są one stosowane w celu uproszczenia budowy skryptów
  - **$x += liczba**
    – zwiększenie wartości zmiennej **$x** o **liczba**;
  - **$x -= liczba**
    – zmniejszenie wartości zmiennej **$x** o **liczba**;
  - **$x *= liczba**
    – mnożenie wartości zmiennej **$x** o **liczba**;
  - **$x /= liczba**
    – dzielenie wartości zmiennej **$x** o **liczba**;

Operatory arytmetyczne

- Zmienna jest strukturą, która (o ile przechowuje liczby) może być traktowana właśnie jak liczba.
- PHP pozwala na wykonywanie na nich standardowych działań dodawania, odejmowania, mnożenia i dzielenia.

+ (*dodawanie*)	$x+$y → suma $x i $y
- (*odejmowanie*)	$x-$y → różnica $x i $y
*** (*mnożenie*)**	$x*$y → iloczyn $x i $y
/ (dzielenie)	$x/$y → iloraz $x i $y
% (*modulo*)	$x%$y → reszta z dzielenia $x i $y

Operatory porównania

- Służą do porównywania wartości stojących po ich lewej i prawej stronie.

Przykład	Nazwa	Opis
$a == $b	równy	Zwraca TRUE, jeśli $a jest równe $b.
$a === $b	identyczny	Zwraca TRUE, jeśli $a jest równe $b i obie zmienne są tego samego typu (tylko w PHP 4).
$a != $b	różny	Zwraca TRUE, jeśli $a nie jest równy $b.
$a <> $b	różny	Zwraca TRUE, jeśli $a nie jest równy $b.
$a !== $b	nie identyczny	Zwraca TRUE, jeśli $a nie jest równy $b lub nie są tego samego typu (tylko w PHP 4).
$a < $b	mniejszy niż	Zwraca TRUE, jeśli $a jest mniejszy od $b.
$a > $b	większy niż	Zwraca TRUE, jeśli $a jest większy od $b.
$a <= $b	nie większy niż	Zwraca TRUE, jeśli $a jest mniejszy lub równy $b.
$a >= $b	nie mniejszy niż	Zwraca TRUE, jeśli $a jest większy lub równy $b.

Inkrementacja i dekrementacja

- Służą podniesieniu lub zmniejszeniu wartości danej zmiennej o 1.

Przykład	Nazwa	Opis
++$a	preinkrementacja	Najpierw zwiększa wartość $a o jeden, a potem zwraca $a.
$a++	postinkrementacja	Najpierw zwraca $a, a potem zwiększa $a o jeden.
--$a	predekrementacja	Najpierw zmniejsza wartość $a o jeden, a potem zwraca $a.
$a--	postdekrementacja	Najpierw zwraca $a, a potem zmniejsza $a o jeden.

Operatory logiczne

- Operatory te wykorzystuje się głównie w instrukcjach warunkowych typu **if**, jeśli umieszczony został więcej niż jeden warunek.

Przykład	Nazwa	Opis
$a and $b	i	Zwraca TRUE, jeśli zarówno $a, jak i $b są TRUE.
$a or $b	lub	Zwraca TRUE, jeśli $a lub $b są TRUE.
$a xor $b	różnica logiczna	Zwraca TRUE, jeśli $a i $b mają róże wartości logiczne.
! $a	negacja	Zwraca TRUE, jeśli $a nie jest TRUE.
$a && $b	i	Zwraca TRUE, jeśli zarówno $a, jak i $b są TRUE.
$a || $b	lub	Zwraca TRUE, jeśli $a lub $b jest TRUE.

Operatory konkatenacji

- Istnieją dwa takie operatory **.** (kropka) oraz **.=** (kropka+znak równości).
  - Pierwszy z nich łączy dwa ciągi z jego lewej i prawej strony, a następnie zapisuje wynik do zmiennej.
  - Drugi operator łańcuchowy (.=) dopisuje wartość ciągu z jego prawej strony do zmiennej z jego lewej

```
<?
        $ciag1 = „tekst_1";
        $ciag2 = " tekst_2";
        $razem_1 = $ciag1.$ciag2;
        $razem_2 = „tekst_1";
        $razem_2 .= $tekst_2;
?>
```

Instrukcje warunkowe

- **Instrukcje warunkowe** są znane ze wszystkich języków programowania.
- Używa się ich do sprawdzenia tego, czy dany warunek jest prawdziwy, i na podstawie wyniku takiego zapytania wykonania określonego fragmentu kodu.
- Jeśli dane sprawdzenie nie przynosi oczekiwanego rezultatu, sprawdzany jest kolejny warunek (i ewentualnie uruchamia przypisany mu blok kodu).
- W wypadku gdy nie jest spełniony żaden warunek, wykonuje się jeszcze inny blok instrukcji.
- Za prawdę uznawane jest wszystko, co ma wartość większą od 0.

Instrukcje warunkowe

```
<?
if (warunek_1) {
instrukcje do wykonania, gdy warunek_1 jest prawdą
}

else if (warunek_2) {
instrukcje do wykonania, gdy drugi warunek_2 jest prawdą
}
//dowolna ilość warunków else if...

else {
instrukcje do wykonania, gdy żaden warunek nie jest
prawdą
}
?>
```

Instrukcje warunkowe - przykład

```
<?
$nastroj = "radosny";
if ($nastroj == "radosny") {
        echo "Hurra, jest wiosna!";
}
$nastroj = "smutny";
if ($nastroj == "radosny") {
        echo "Hurra, jest wiosna!";
} else {
        echo "Może i jest wiosna ale jest zimno.";
}
?>
```

Pętla FOR

⇨ **Pętla FOR** jest używana tylko wtedy, gdy zachodzi potrzeba wykonania jakiegoś kodu określoną liczbę razy (założoną z góry lub pochodzącą ze zmiennej).

⇨ Składnia tej pętli przedstawia się następująco:

```
<?
for (inicjalizacja zmiennych; sprawdzenie warunku;
modyfikacja zmiennych) { instrukcje do wykonania }
?>
```

```
<?
for( $i=10; $i<20 ; $i++ ) {
echo "$i<span style=\"font-size:".$i."px\"> tekst
</span><br />"; }
?>
```

Pętla WHILE

⇨ **Pętla WHILE** jest wykonywana tak długo, jak długo jest prawdziwy warunek zdefiniowany po słowie kluczowym.

```
<?
while (warunek) {
instrukcje do wykonania
}
?>
```

```
<?
$i=10;
while ($i<20) {
echo "$i<span style=\"font-size:".$i."px\"> tekst
</span><br />";
$i++; }
?>
```

Pętla DO … WHILE

⇨ **Pętla DO … WHILE** to specyficzna odmiana pętli WHILE, bo jeśli we WHILE warunek jest na starcie fałszywy, to pętla ani razu nie wykona bloku instrukcji.

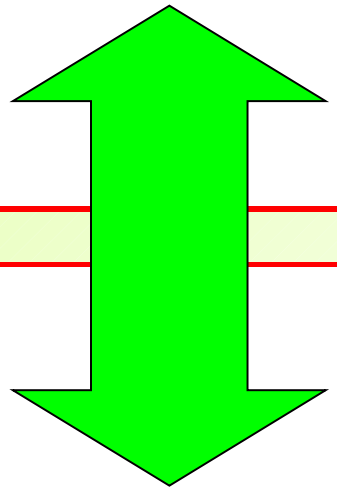
⇨ W wypadku użycia DO…WHILE pętla wykona blok instrukcji przynajmniej raz (nawet wtedy, gdy warunek jest od początku fałszywy).

```
<?
do {
instrukcje do wykonania
} while (warunek);
?>
```

Pętla FOREACH

- **Pętla FOREACH** została zaczerpnięta z języka Perla.
- Ułatwia ona obsługę tablic i tablic asocjacyjnych.

```
<?
for($i=0;$i<sizeof($kolory);$i++) {
echo „Wybrany kolor to: ".$kolory[$i]
}
?>
```



```
<?
foreach ($kolory as $kolor) {
echo „Wybrany kolor to: ".$kolor;
}
?>
```

Instrukcja SWITCH

⇨ **SWITCH** jest rodzajem skondensowanej instrukcji warunkowej, którą zazwyczaj zastępujemy rozbudowane i wielokrotne użycia ELSE IF.

```
<?
switch ($miasto) {
case 'warszawa': echo "Pochodzisz ze stolicy?"; break;
case 'hel': echo "Mieszkasz nad morzem?"; break;
case 'sanok': echo "A może w Bieszczadach?"; break;
default: echo "Miasto nierozpoznane.";
}
?>
```

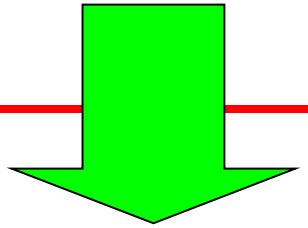
Przerwanie wykonania pętli

- Służą do tego instrukcje **BREAK** oraz **CONTINUE**.
- Pierwsza z nich powoduje przerwanie wykonywania pętli, a co za tym idzie – dalszej części zawartego w niej kodu.
- CONTINUE powoduje natomiast przerwanie aktualnej iteracji (przebiegu) pętli i powrót do jej początku.

Instrukcje alternatywne

- Istnieją też składnie alternatywne do przedstawionych powyżej.
- Pierwsza z nich to skrócona wersja instrukcji IF.

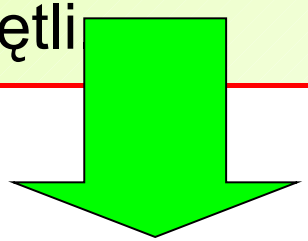
```
<?
if (warunek) {
instrukcje jeśli warunek jest prawdziwy}

else {
 instrukcje w przeciwnym wypadku}
?>
```



```
<?
((warunek)? instrukcje jeśli warunek jest
prawdziwy : instrukcje w przeciwnym wypadku)
?>
```

Instrukcje alternatywne

- Kolejną możliwością jest niewykorzystywanie nawiasów klamrowych w instrukcjach IF, FOR, SWITCH, WHILE oraz FOREACH.
- Zamiast klamry otwierającej należy umieścić dwukropek, a na końcu słowo ENDIF, ENDSWITCH bądź inny END... – odpowiedni do użytej pętli



```
<?
if (warunek) :
instrukcje jeśli warunek jest prawdziwy
else:
instrukcje w przeciwnym wypadku
endif;
?>
```

PHP - tablice

- **Tablice** to typy zmiennych, które służą do przechowywania innych zmiennych.
- Używanie tablic pomaga np. w przeglądaniu wpisów pobranych z bazy danych lub odczytanej listy plików z folderu na serwerze, gdyż można się do nich odwoływać po ich indeksach (liczby całkowite) lub identyfikatorach tekstowych (w tablicach asocjacyjnych).
- Elementem tablicy może być dowolna zmienna, lub inna tablica.

```
<?
$tablica[0] = 8;
$tablica[1] = 9;
$tablica[2] = 1;
?>
```

```
<?
$tablica[] = 8;
$tablica[] = 9;
$tablica[] = 1;
?>
```

PHP – tablice asocjacyjne

- **Tablice asocjacyjne** to odmiany tablic, w których zamiast indeksów liczbowych występują indeksy tekstowe.
- Najczęstsze zastosowanie tej struktury danych prezentuje poniższy przykład - tablica reprezentuje konkretną osobę, a wiersze to jej dane osobowe.

```
<?
echo $kowalski["imie"] = "Jan";
echo $kowalski["miasto"] = "Warszawa";
echo $kowalski["ulica"] = "Polna";
echo $kowalski["wiek"] = 19;
echo $kowalski["kawaler"] = true;
?>
```

Przeglądanie tablic

- W przypadku zwykłej tablicy (gdy znana jest liczba jej elementów) wystarczy użyć pętli FOR i następującej konstrukcji:

```
<?
for ($i=0;$i<10;$i++) { echo $tablica[$i]; }
?>
```

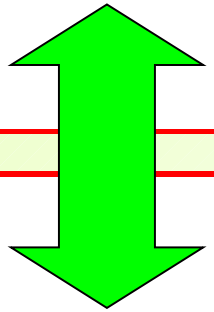
- Jeśli liczba elementów tablicy nie jest znana, to należy posłużyć się funkcją **sizeof()**, podając jako argument

```
<?
for ($i=0;$i< sizeof($tablica);$i++) { echo $tablica[$i]; }
?>
```

Przeglądanie tablic asocjacyjnych

Można w tym celu wykorzystać pętlę FOREACH lub pętlę WHILE oraz funkcje **list( )** i **each( )**.

```
<?
foreach ($tablica as $klucz => $wartosc) {
echo "$klucz => $wartosc <br />";
}
?>
```



```
<?
while( list($klucz, $wartosc) = each($tablica) )
echo "$klucz => $wartosc<br />";
}
?>
```

przypisuje listę zmiennych do funkcji zwracającej tablicę wyników

zwraca bieżącą parę klucza i wartości z tablicy oraz przesuwa kursor tablicy

Zamiana tekstu na tablicę

- Zdarza się, że ciąg znaków trzeba zamienić na tablicę (chociażby przy odczytywaniu danych z pliku).
- Wykorzystuje się w tym celu funkcję **explode()**, która rozdziela określony ciąg znaków i tworzy z powstałych elementów tablicę.
- Funkcja pobiera dwa argumenty: ciąg znaków lub znak, który stanowi element rozdzielający, a także zmienną zawierającą ciąg, który chcemy rozdzielić.

```
<?
$wpisy = explode("|", $odczyt);
?>
```

Wywołanie zwróci tablicę o nazwie $wpisy z pojedynczymi elementami, które były w oryginalnym ciągu rozdzielone symbolem |.

Zamiana tablicy na tekst

- Elementy tablicy można również połączyć określonym znakiem (aby je np. zapisać w pliku).
- Służy do tego funkcja **implode()** - pobiera ona dwa argumenty: pierwszy to ciąg, którymi mają być połączone elementy tablicy, a drugi to nazwa zmiennej tablicy z danymi.

```
<?
$doZapisu = implode("|", $wpisy);
?>
```

PHP – sortowanie tablic

Do sortowania tablic – zwykłych, jak i asocjacyjnych napisano wiele funkcji, najważniejsze z nich to:

- **asort()** – sortuje rosnąco tablice asocjacyjne według wartości kluczy, zachowując przypisanie kluczy do wartości,
- **arsort()** – sortuje malejąco tablice asocjacyjne według wartości kluczy, zachowując przypisanie kluczy do wartości,
- **ksort()** – sortuje rosnąco tablice asocjacyjne według kluczy, zachowując przypisanie kluczy do wartości,
- **krsort()** – sortuje malejąco tablice asocjacyjne według kluczy, zachowując przypisanie kluczy do wartości,

PHP – sortowanie tablic

- **sort()** – sortuje rosnąco zwykłe tablice,
- **rsort()** – sortuje malejąco zwykłe tablice,
- **uasort()** – funkcja sortująca tablice asocjacyjne za pomocą zdefiniowanej przez użytkownika funkcji porównującej elementy (nazwa funkcji jest podawana za pomocą drugiego parametru),
- **usort()** – funkcja sortująca zwykłe tablice za pomocą funkcji zdefiniowanej przez użytkownika,
- **uksort()** – funkcja sortująca tablice asocjacyjne według klucza za pomocą funkcji zdefiniowanej przez użytkownika.

PHP – inne operacje na tablicach

- **array_change_key_case** (array wejście [, int wielkość])

  zmienia klucze w tablicy wejście, tak aby były one pisane tylko dużymi lub tylko małymi literami - zmiana zależy od ostatniego opcjonalnego parametru case (można do niego przekazać jedną z dwóch stałych: CASE_UPPER lub CASE_LOWER - domyślną wartością jest CASE_LOWER).

- **array_flip** (trans)

  zwraca tablicę w odwróconym porządku, tzn. klucze z tabeli trans stają się wartościami, a wartości trans stają się kluczami.

- **array_fill** (indeks_początkowy, num, wartość)

  wypełnia tablicę wartością wartość, począwszy od indeksu indeks_początkowy przez num kolejnych elementów tablicy,

PHP – inne operacje na tablicach

- **array_pop** (tablica)

  zdejmuje i zwraca ostatnią wartość tablicy tablica, skracając tę tablicę o jeden element. Jeśli tablica jest pusta (lub nie jest tablicą), zwracana jest wartość NULL,

- **array_push** (tablica, wartosc [, wartosc ...])

  traktuje zmienną tablica jako stos i wstawia przekazane parametry na koniec podanej tablicy - długość parametru tablica zwiększa się o liczbę przekazanych wartości. Funkcja zwraca nową liczbę elementów tablicy,

- **array_shift** (tablica)

  usuwa pierwszą wartość parametru tablica i zwraca go, skracając tę tablicę o jeden element i przesuwając wszystkie pozostałe elementy w dół. Jeśli tablica jest pusta (lub nie jest tablicą), zwracana jest wartość NULL,

PHP – inne operacje na tablicach

- **array_unshift** (tablica, wartość, [wartosc...])

  wstawia jeden lub więcej przekazanych jako parametry elementów na początek tablicy tablica. Zauważmy, że lista elementów jest wstawiana jako całość, więc elementy zostają w takim samym porządku. Funkcja zwraca nową liczbę elementów w tablicy tablica,

- **array_search** (igła, stóg_siana [, ścisły])

  przeszukuje stóg_siana w poszukiwaniu parametru igła i zwraca odpowiedni klucz, jeśli został on znaleziony, lub FALSE w przeciwnym wypadku. Jeżeli trzeci parametr ścisły jest ustawiony na TRUE, to array_search() porówna także typy parametru igła z tymi z parametru stóg_siana.

PHP – tryb dostęp do pliku

- Wszystkie funkcje obsługujące pliki jako pierwszy parametr pobierają zmienną – tzw. uchwyt do pliku.

- Jest to specjalna nazwa zwracana przez funkcję otwierającą plik – **fopen()** – która jednoznacznie identyfikuje dany zbiór.

- Funkcji **fopen()** pobiera dwa parametry:
  - pierwszy to nazwa pliku do otwarcia,
  - drugi to tryb otwarcia, określający cel w jakim dokonano otwarcia pliku.

PHP – tryb dostęp do pliku

Istnieje pięć takich trybów:

- **r** plik tylko do odczytu (wewnętrzny wskaźnik pliku umieszczany jest na początku zbioru),
- **r+** plik do odczytu i zapisu (wewnętrzny wskaźnik pliku umieszczany jest na początku zbioru),
- **w** plik tylko do zapisu (wewnętrzny wskaźnik pliku umieszczany jest na końcu zbioru),
- **w+** plik do odczytu i do zapisu (wewnętrzny wskaźnik pliku umieszczany jest na końcu pliku),
- **a** plik tylko do zapisu (wewnętrzny wskaźnik pliku umieszczany jest na końcu pliku).

PHP – otwarcie i zamkniecie pliku

Aby otworzyć plik do odczytu, wystarczy wpisać kod:

```
<?
$plik = fopen("dane/imiona.txt" , "r");
?>
```

Po zakończeniu używania pliku można (ale nie trzeba) go zamknąć funkcją **fclose**(wskaznik_do_pliku)

```
<?
fclose($plik);
?>
```

Wewnętrzny wskaźnik pliku

- Wewnętrzny wskaźnik pliku określa, skąd w pliku mają być odczytywane dane lub gdzie mają być zapisywane.
- Wskaźnik przesuwa się automatycznie dalej po każdej procedurze odczytania określonej porcji danych z pliku.
- Do ustawienia wskaźnika w określonym miejscu pliku można użyć funkcji **fseek().**
- Funkcja pobiera trzy argumenty:
  - pierwszy to uchwyt do pliku,
  - kolejny to przesunięcie,
  - trzeci (opcjonalny) to rodzaj przesunięcia.

Wewnętrzny wskaźnik pliku

Dostępne są trzy wartości przesunięcia:

- SEEK_SET ustawia wskaźnik na pozycję określoną poprzez przesunięcie (domyślne),
- SEEK_CUR ustawia wskaźnik na pozycję równą aktualnej i uwzględnia przesunięcie,
- SEEK_END ustawia wskaźnik na koniec pliku i uwzględnia przesunięcie - aby ustawić wskaźnik w konkretnej odległości

(liczby znaków) od końca zbioru, przesunięcie musi być ujemne.

Odczyt z pliku

Jest kilka możliwości odczytu danych z pliku:

① Pierwsza to odczyt po kolei po jednym znaku.

- Służy do tego funkcja **fgetc()**.
- Wystarczy jako argument podać tylko uchwyt do pliku.
- Funkcja zwraca jeden odczytany znak lub false, jeśli osiągnięty został koniec zbioru.

```
<?
while (false !== ($znak = fgetc($file))) {
echo "Aktualny znak to $znak <br />";
}
?>
```

Odczyt z pliku

② Kolejną możliwością jest odczyt linijka po linijce.

- Służy do tego funkcja **fgets()** .
- Jako parametry należy podać uchwyt do pliku i opcjonalnie maksymalną długość odczytanej linii.
- Czytanie linii kończy się, gdy funkcja dojdzie do końca linii (znak nowej linii jest dołączany do zwróconej wartości) lub do końca pliku.

```
<?
while (false !== ($linia = fgets($file))) {
echo "Aktualna linia to $linia";
}
?>
```

Odczyt z pliku

③ Można też odczytać od razu cały plik.

- Służy do tego funkcja **fread()** .
- Jej argumentami są: uchwyt do pliku i liczba znaków, jaka ma być odczytana.
- Często jako liczbę znaków podaje się całkowitą długość pliku, obliczoną funkcją **filesize(**nazwa_pliku**)**.

```
<?
$plik = fopen("plik.txt", "r");
$dane = fread($plik, filesize("plik.txt"));
?>
```

UWAGA:

Jeśli podana ilość danych do odczytania jest mniejsza niż długość pliku, to po ponownym wywołaniu tej funkcji zwróci ona dane, od momentu gdzie zakończyło się poprzednie czytanie z pliku.

Odczyt z pliku

④ Kolejna możliwość to odczyt całego pliku za pomocą tylko jednej linijki kodu:

```
<?
$dane = fread(fopen("plik.txt", "r"), filesize("plik.txt"));
?>
```

```
<?
$dane = file_get_contents("plik.txt");
?>
```

⑤ Kolejną (ostatnią) metodą odczytywania zawartości pliku jest przeniesienie jego zawartości do tablicy, gdzie każdy wpis odpowiada jednej linii z pliku - służy do tego funkcja **file()**, w której wystarczy podać tylko nazwę pliku

Zapis do pliku

① Do zapisu do pliku wykorzystywana jest funkcja **fwrite()**.

- Problem z zapisywaniem jest taki, że nie ma możliwości zapisu na końcu czy w środku pliku (można dopisywać tylko na początku).

```
<?
$file = fopen("plik.txt", "r");
$dane = fread($file, filesize("plik.txt"));
fclose($file);
$noweDane = "Coś do dodania na początku pliku";
$noweDane .= $dane;
$file = fopen("plik.txt", "w");
fwrite($plik, $noweDane);
fclose($file);
?>
```

PHP – blokowanie dostępu do pliku

- Podczas używania plików w większym serwisie (o dużej liczbie odwiedzin) może się zdarzyć, że w tym samym czasie dwa procesy będą coś próbowały zapisywać do pliku.

- Jeśli odbędzie się to dokładnie w tym samym czasie, to z zawartością pliku mogą się zdarzyć różne nieprzewidywalne rzeczy, aby temu zapobiec stosuje blokady pliku.

- W PHP funkcjonują dwa rodzaje blokad:
  - blokada dzielona – może być założona przez wiele procesów naraz podczas odczytu,
  - blokada wyłączna – zakładana podczas zapisu do pliku tylko przez jeden proces.

PHP – blokowanie dostępu do pliku

Aby założyć blokadę, należy użyć funkcji **flock()**, która pobiera dwa argumenty: uchwyt do pliku i rodzaj blokady.

Dostępne są trzy rodzaje blokady:

- LOCK_SH – zakłada blokadę dzieloną (do odczytu),
- LOCK_EX – zakłada blokadę wyłączną (do zapisu),
- LOCK_UN – zdejmuje blokadę z pliku.

Jeśli zakładanie blokady się powiedzie, to funkcja zwraca wartość true, w przeciwnym wypadku zwracana jest wartość false.

PHP – blokowanie dostępu do pliku

```
<?
$file = fopen("dane\plik.txt", "w+");
if (flock( $file, LOCK_EX)) {
fwrite($file, "Wartość dopisana do pliku");
flock( $file, LOCK_UN);
} else {
echo "Błąd - plik zablokowany";
}
fclose($file);
?>
```

zdjęcie blokady

zakładanie blokady wyłącznej

PHP – informacje o pliku

W PHP istnieje też kilka funkcji, które zwracają informacje o pliku - należą do nich:

- fileatime(nazwa_pliku) – zwraca datę i czas ostatniego odczytu pliku podane w formacie timestamp,
- filemtime(nazwa_pliku) – zwraca datę i czas ostatniej modyfikacji pliku podane w formacie timestamp,
- filegroup(nazwa_pliku) – zwraca liczbowy identyfikator grupy, do której należy właściciel pliku,
- fileowner(nazwa_pliku) – zwraca identyfikator właściciela pliku,
- fileperms(nazwa_pliku) – zwraca prawa dostępu do pliku,
- filesize(nazwa_pliku) – zwraca wielkość pliku w bajtach.

PHP – informacje o pliku

Oprócz tego istnieje też kilka funkcji zwracających wartości true lub false.

- is_dir(nazwa_pliku) – informuje o tym, czy zbiór jest katalogiem,
- is_executable(nazwa_pliku) – informuje o tym, czy plik jest wykonywalny,
- is_file(nazwa_pliku) – informuje o tym, że plik istnieje i jest zwykłym plikiem,
- is_readable(nazwa_pliku) – informuje o tym, czy plik można odczytać,
- is_writeable(nazwa_pliku) – informuje o tym, czy plik można zapisywać,
- is_uploaded_file(nazwa_pliku) – informuje o tym, czy plik został wysłany z formularza.

Operacje na plikach i katalogach

❶ W PHP kopiowanie plików odbywa się przy pomocy funkcji **copy()**, która zwraca wartość true, jeśli plik zostanie poprawnie skopiowany, lub false, jeżeli nie.

```
<?
copy("tymczasowy/plik.txt", "dane/plik.txt");
?>
```

❷ Aby zmienić nazwę zbioru, należy użyć funkcji o nazwie **rename()**. Może ona też służyć do kopiowania plików

```
<?
if (!rename("/tmp/tmp_file.txt", "/home/user/my_file.txt");)
echo "Błąd podczas kopiowania";
?>
```

Operacje na plikach i katalogach

❸ Do usuwanie plików służy funkcja o nazwie **unlink()**, w której argumentem jest nazwa usuwanego pliku. Operacja wymaga by uprawnienia zbioru pozwalały na jego usunięcie, w przeciwnym razie zostanie wyświetlony błąd.

❹ Aby utworzyć katalog, należy wywołać funkcję **mkdir()** i jako parametry podać kolejno: nazwę katalogu do utworzenia i jego prawa dostępu (np. 0777).

❺ Do usuwania katalogu służy do tego funkcja **rmdir()**, której argumentem jest nazwa folderu przeznaczonego do usunięcia. Warunkiem koniecznym, by operacja się powiodła, jest to, aby usuwany katalog był pusty.

Operacje na plikach i katalogach

6. Do przeglądania zawartości katalogów służy mechanizm pseudoobiektowy.
   - Przeglądanie rozpoczyna funkcja **dir(**nazwa_folderu**)**, która zwraca obiekt-uchwyt do katalogu.
   - Kolejne pozycje z katalogu pobierane są za pomocą metody **read()** .
   - Pracę z folderem kończy się metodą **close()**.

```
<?
$dir=dir("logs");
while($entry=$dir->read()) {
echo "Kolejna pozycja z tego folderu to $entry<br />";
}
$dir->close();
?>
```

Prawa dostępu

- Są to zezwolenia dla konkretnych użytkowników lub całych ich grup na wykonanie operacji na danym pliku.
- Każdemu plikowi i katalogowi w systemie można przypisać trzy komplety praw:
  - pierwszy z tych kompletów dotyczy właściciela pliku,
  - drugi grupy użytkowników,
  - a trzeci użytkowników, którzy ani nie są właścicielami zbioru, ani nie należą do grupy.
- Pojedynczy komplet to suma praw:
  - uruchomienia lub w wypadku katalogu: otwarcia (wartość 1),
  - odczytu (wartość 4)
  - i zapisu (wartość 2).

Prawa dostępu

- W PHP do ustawiania praw dostępu służy funkcja **chmod(**$nazwa_pliku, $tryb**)**, gdzie drugi parametr to prawa dostępu zapisane w formacie ósemkowym.
- Inne funkcje przydatne przy pracy z systemem zabezpieczeń to:
  - **chown(**$nazwa_pliku, $user**)**, zmieniająca właściciela pliku,
  - **chgrp(**$nazwa_pliku, $grupa**)**, zmieniająca grupę.
- Ustawienie odpowiednich praw dostępu to sprawa podstawowa przy tworzeniu skryptów zapisujących coś do pliku lub operujących na katalogach - bardzo często wyświetla się błąd „Permission denied”, którego poprawienie to właśnie kwestia ustawienia odpowiednich praw dostępu

PHP – złączenia zewnętrznych plików

Cały skrypt nie zawsze musi się znajdować w jednym pliku. Jeśli budujemy rozbudowany serwis, na pewno są w nim elementy, które powtarzają się we wszystkich zbiorach. Nie muszą być one umieszczane w każdym pliku ze skryptem – wystarczy stworzyć jeden plik z daną funkcją lub klasą i załączać go do pozostałych.

W PHP służą temu cztery funkcje:

- **include()**
- **require()**
- **include_once()**
- **require_once()**

PHP – złączenia zewnętrznych plików

- Plik załączany poprzez **include()** jest wczytywany i załączany, gdy wykonywana jest linia z instrukcją, a wszystkie zmienne dostępne we wczytanym pliku będą dostępne w pliku głównym dopiero od miejsca, gdzie znajduje się instrukcja include().

- Dla odmiany **require()** od razu próbuje wczytać żądany plik, nawet jeśli jego wywołanie znajduje się w instrukcji warunkowej (może nie zostać wywołane).

- Funkcje **include_once()** i **require_once()** dodano w PHP 4.0.1 - działają podobnie jak include() i require(), z tą jedną różnicą, że nawet wielokrotne wywołanie includowania danego pliku poprzez include_once() lub require_once() nie spowoduje wczytania go kilkakrotnie.

PHP – złączenia zewnętrznych plików

```
<?
//to jest poprawny zapis:
if ($a>5) {
include ("plik1.php");
}
else {
include ("plik2.php");
}
?>
```

```
<?
require_once("a.php"); //wczyta a.php
require_once("A.php"); //a Windows znowu wczyta a.php
?>
```

PHP – przekazywanie zmiennych

☒ Istnieją dwa sposoby na przekazanie zmiennej między dwoma plikami PHP – metody **GET** i **POST**.

☒ **Metoda GET** polega na wywołaniu pliku PHP z

```
?zmienna1=wart_1&zmienna2=wart_2&zmienna3=wart_3
```

Należy pamiętać, że wysyłanie zmiennych widocznych w URL prowadzi do sytuacji, w której ktoś może łatwo podmienić ich wartość i np. odczytać plik z hasłami umieszczony na serwerze.

☒ **Metoda POST** służy raczej do przekazywania dużej liczby zmiennych przy zachowaniu podstawowych zasad poufności - nazwy zmiennych i ich wartości nie są bowiem widoczne, ponieważ są wysyłane jako część nagłówka pliku.

PHP – przekazywanie zmiennych

☒ Zmienne przekazane jako POST lub GET trafiają do tablic superglobalnych – są to tablice asocjacyjne w których nazwą klucza jest nazwa zmiennej, a wartością wartość tej zmiennej.

☒ Dostępne w PHP od wersji 4.10 tablice superglobalne to:

- **$GLOBALS** – przechowuje wszystkie zmienne globalne w skrypcie,
- **$_POST** – wszystkie zmienne wysłane metodą POST,
- **$_GET** – wszystkie zmienne wysłane metodą GET,
- **$_SERVER** – przechowuje zmienne tworzone przez serwer lub przez środowisko uruchamiania skryptu,
- **$_ENV** – przechowuje zmienne tworzone przez środowisko wykonywania skryptu,

PHP – przekazywanie zmiennych

- **$_COOKIE** – przechowuje zmienne pochodzące z ciasteczek,
- **$_REQUEST** – przechowuje zmienne dostarczane przez różne mechanizmy wejścia ($_GET, $_POST, $_ENV, $_COOKIE) razem – tablica nie jest bezpieczna, gdyż nie wyszczególnia tego, skąd pochodzi dana zmienna,
- **$_SESSION** – przechowuje zmienne pochodzące z sesji,
- **$_FILES** – przechowuje pliki wgrane na serwer poprzez formularz.

PHP – przekazywanie zmiennych

Formularz wysyłający dane o użytkowniku do innego pliku.

a) plik formularz.html:

```
<form action="dane.php" method="POST">
<input type="text" name="imie" size="15" /><br />
<input type="text" name="nazwisko" size="16" /><br />
<input type="radio" name="plec" value="m" /> Mężczyzna<br />
<input type="radio" name="plec" value="k" /> Kobieta<br />
<input type="submit" value="Wyślij dane" /><br />
</form>
```

b) plik dane.php

```
<?
echo "Witaj ".$_POST['imie']." ".$_POST['nazwisko']."<br />";
echo "Jesteś ".(($_POST['plec']=="m")?"mężczyzną":"kobietą")
." <br />";
?>
```

PHP – ciasteczka

Cookies, czyli tzw. **ciasteczka**, to małe pliki tekstowe przechowywane na komputerze odbiorcy, w których przetrzymywane są informacje np. o dacie jego ostatniej wizyty lub jego nazwie użytkownika na forum.

PHP ma możliwość zapisu ciasteczek dzięki funkcji **setcookie()** - argumenty, jakie należy podać, to:

- Czas wygaśnięcia to czas, po którym ciastko nie będzie dostępne do odczytu - jeśli czas nie zostanie podany, to ciastko będzie ważne do zamknięcia przeglądarki.
- Ścieżka i domena określa kolejno ścieżkę na serwerze i domenę, z których ciastko może zostać odczytane.
- Bezpieczeństwo (gdy 1, to ciastko będzie można odczytać, tylko przy szyfrowanym połączeniu (SSL).

PHP – ciasteczka

```
<?
$iloscWejsc++;
setcookie("iloscWejsc", $iloscWejsc);
?>
<HTML>
<BODY>
Odwiedzasz tą stronę już po raz
<?
echo($iloscWejsc);
?>
</BODY>
</HTML>
```

PHP – funkcje

- Często pisząc rozbudowany skrypt, wykorzystuje się pewne fragmenty kodu (np. pobranie porcji danych z bazy i odpowiednie przetworzenie jej) wiele razy.
- Zamiast wielokrotnie pisać ten sam kod, można stworzyć funkcję, czyli coś w rodzaju podprogramu wykonującego odpowiednie zadania.

```
function nazwa_funkcji($argument1, $argument2)
        { /treść funkcji }
```

```
<?
function f($a, $b)      { $a += $b;   echo($a); }
// wywołanie funkcji
f(7, 2);
?>
```

PHP – funkcje – zasięg zmiennej

- Kiedy zmienna jest zadeklarowana poza funkcją, jej wartość nie będzie widoczna w funkcji.
- Jej wartość będzie widoczna w funkcji jeśli wcześniej użyto deklarację **global**.

```
<?
$a=7; $b=2;
global $a;
function f() {
echo($a);
echo($a);
}
f();
?>
```

wywołanie funkcji **f** zwróci tylko wartość zmiennej $a

PHP – data i czas

- W PHP istnieje bardzo dużo funkcji pozwalających na swobodne operowanie datą i czasem.
- Podstawowym formatem daty jest **timestamp**, czyli liczba sekund, która upłynęła od 1 stycznia 1970 roku.
- Najprostszą akcją jest uzyskanie aktualnej daty i czasu w formacie timestamp jest użycie funkcji **time()**.

```
<?
$teraz = time();
?>
```

- Funkcja **microtime()** zwraca dane w takiej postaci, że część milisekundowa jest oddzielona spacją od części sekundowej

PHP – data i czas

Aby sprawdzić, czy data wprowadzona np. w formularzu jest poprawna, należy posłużyć się funkcją **checkdate()**, która pobiera ona kolejno miesiąc, dzień i rok (kolejność zapisu daty w USA) i zwraca wartość TRUE, jeśli data jest poprawna, lub FALSE w wypadku błędu.

```
<?
if (checkdate($_POST[month], $_POST[day],
$_POST[year])) {
//data jest poprawna - dalsze instrukcje
}
else {
//data niepoprawna - wprowadz jeszcze raz
Header("Location:podaj_date.php?blad=1");
}
?>
```

PHP – data -> timestamp

- Datę zapisaną w formularzu w formacie: *dzień, miesiąc, rok, godziny, minuty i sekundy* można zamienić na format timestamp używając funkcji **mktime()**.

- Jako kolejnych argumentów tej funkcji należy podać: godzinę, minuty, sekundy, miesiąc, dzień i rok.

- Jako ostatni (niewymagany) parametr można podać 1 (jeśli jest to czas letni), 0 (jeżeli jest to czas zimowy) lub -1 bądź nie podawać niczego (wtedy interpreter PHP będzie sam zgadywał, jaki to czas).

```
<?
$time = mktime(23, 24, 11, 12, 4, 2009);
?>
```

PHP – data -> timestamp

Funkcja **mktime()** nadaje się także do poprawiania błędów w datach i wyliczania ostatnich dni miesiąca.

```
<?
$time = mktime(1, 1, 1, 12, 32, 2009);
$time = mktime(1, 1, 1, 13, 1, 2009);
$time = mktime(1, 1, 1, 1, 1, 2009);
$time = mktime(1, 1, 1, 1, 1, 2009);
?>
```

Wszystkie powyższe przykłady utworzą datę 1 stycznia 2010, a błędy typu 32 grudnia czy 1 dnia 13 miesiąca roku zostaną automatycznie poprawione.

Aby wyliczyć ostatni dzień danego miesiąca, należy podać jako argument dzień 0 (zero), a numer miesiąca o jeden wiekszy.

PHP – timestamp -> data

- Przekształcenie daty z wewnętrznego formatu timestamp do postaci dającej się zinterpretować przez użytkownika umożliwia funkcja **date()**.

- Jako pierwszy argument pobiera ona format, w którym zwrócona ma być data, jako drugi zaś (opcjonalnie) datę w formacie timestamp.

- Funkcja zwraca datę sformatowaną zgodnie z szablonem w argumencie format - w szablonie podaje się specjalne znaki, które są później zamieniane na odpowiednie elementy daty lub czasu
  - a – „am” lub „pm”,
  - A – „AM” lub „PM”,
  - B – czas internetowy Swatcha,

PHP – timestamp -> data

- d – dzień miesiąca, dwie cyfry z zerem na początku, tzn. od „01” do „31”,
- D – dzień tygodnia, tekst, trzy litery, np. „Fri”,
- F – miesiąc, tekst, pełna nazwa, np. „January”,
- g – godzina, format 12-godzinny bez zera na początku, tzn. od „1” do „12”,
- G – godzina, format 24-godzinny bez zera na początku, tzn. od „0” do „23”,
- h – godzina, format 12-godzinny z zerem na początku, tzn. od „01” do „12”,
- H – godzina, format 24-godzinny z zerem na początku, tzn. od „00” do „23”,
- i – minuty; tzn. od „00” do „59”,
- I – „1”, jeśli czas oszczędzania światła słonecznego (w czas letni); „0”, jeżeli czas standardowy (zimowy),

PHP – timestamp -> data

- j – dzień miesiąca bez zera na początku,
- l – dzień tygodnia, tekst, pełna nazwa, np. „Friday”,
- L – „1”, jeśli rok przestępny; „0” w przeciwnym wypadku,
- m – miesiąc, tzn. od „01” do „12”,
- M – miesiąc, tekst, trzy litery, np. „Jan”,
- n – miesiąc bez zera na początku, tzn. od „1” do „12”,
- O – różnica w stosunku do czasu Greenwich, np. „+0200”,
- r – data sformatowana według RFC 822, np. „Thu, 21 Dec 2000 16:01:07 +0200” (dodane w PHP 4.0.4),
- s – sekundy, np. od „00” do „59”,
- S – standardowy angielski sufiks liczebnika porządkowego, dwie litery, tzn. „st”, „nd”, „rd” lub „th”,
- t – liczba dni w danym miesiącu, tzn. od „28” do „31”,
- T – strefa czasowa ustawiona na tej maszynie, np. „EST” lub „MDT”.

PHP – timestamp -> data

- U – liczba sekund od uniksowej Epoki (1 stycznia 1970 00:00:00 GMT),
- w – dzień tygodnia, liczbowy, tzn. od „0” (Niedziela) do „6” (Sobota),
- W – numer tygodnia w roku według ISO-8601, tydzień zaczyna się w poniedziałek (dodane w PHP 4.1.0),
- Y – rok, cztery liczby, np. „1999”,
- y – rok, dwie liczby, np. „99”,
- z – dzień roku, tzn. od „0” do „365”,
- Z – ofset strefy czasowej w sekundach (tzn. pomiędzy „-43200” a „43200”). Ofset dla stref czasowych na zachód od UTC (południka zero) jest zawsze ujemny, a dla tych na wschód od UTC jest zawsze dodatni.

PHP – timestamp -> data

Przykłady użycia funkcji **date():**

```
<?
$today = date("F j, Y, g:i a" );
                    // March 10, 2001, 5:16 pm
$today = date("m.d.y");
                    // 03.10.01
$today = date("j, n, Y");
                    // 10, 3, 2001
$today = date("D M j G:i:s T Y");
                    // Sat Mar 10 15:16:08 MST 2001
$today = date("H:m:s \m \i\s\ \m\o\t\h");
                    // 17:03:17 m is month
$today = date("H:i:s");
                     // 17:16:17
?>
```

PHP – sesje

- Dzięki sesjom można przechowywać pewne dane (takie jak identyfikator użytkownika czy jego login) podczas następujących po sobie wywołań strony.
- Działa to podobnie jak ciasteczka bez ustawionego czasu wygaśnięcia (expire) – przy czym dostęp do danych znika przy wylogowaniu się (dzięki specjalnemu przyciskowi na stronie) lub zamknięciu wszystkich okien przeglądarki.
- Sesje od ciastek odróżnia również to, że wszystkie dane są przechowywane nie u klienta, ale na serwerze.
- Na komputerze gościa trzymany jest jedynie identyfikator sesji, który znajduje się w ciasteczku lub przekazywany jest przez adres strony metodą GET.

PHP – sesje

- Sesje rozpoczyna się, uruchamiając na stronie funkcję **session_start()**, która sprawdza czy nie istnieje już rozpoczęta sesja.
- Jeśli tak, to pobiera identyfikator sesji z URL lub z ciastka i odtwarza wszystkie zmienne przechowywane w sesji, a jeżeli nie, to tworzy nową sesję i nadaje unikatowy identyfikator.
- Dostęp do zmiennych sesyjnych odbywa się poprzez obecną w PHP w wersji od 4.0.6 tablicę superglobalną $_SESSION.

PHP – sesje

```
<?
session_start();
if (isset($_SESSION['licznik']))
        $_SESSION['licznik']++;
else
        $_SESSION['licznik']=1;
echo "Oglądasz tą stronę już ".$_SESSION['licznik']." raz";
?>
```

Jeśli zmienna zapisana nie jest już potrzebna, to należy użyć funkcji **unset()** , a zostanie ona „wymazana” z sesji.

```
<?
session_start();
unset($_SESSION['licznik']);
?>
```

PHP – dynamiczna grafika

- Dzięki użyciu obecnej domyślnie w PHP biblioteki GD istnieje możliwość dynamicznego tworzenia plików graficznych.
- Grafika taka jest tworzona przez parser PHP, a następnie wysyłana do przeglądarki tak jak zwykły plik graficzny.
- Dzięki GD można utworzyć grafikę w różnych formatach, m.in. JPG, GIF, WBMP i PNG, przy czym zalecane jest używanie właśnie PNG, który wypiera GIF-y z sieci.
- Dynamiczną grafikę wstawia się do dokumentu tradycyjnie, używając znacznika <img>.

```
<?
echo"<img src=\"licznik.php?wizyty=".$licz."\" />";
?>
```

PHP – dynamiczna grafika

Możliwe jest nawet uruchamianie skryptu tworzącego grafikę z parametrami podanymi jako zmienne GET.

Najważniejsze jest ustawienie Content-type w nagłówku, aby przeglądarka mogła odpowiednio zinterpretować plik.

```
<?
header("Content-type: image/png");
?>
```

W zależności od założonego typu pliku należy ustawić odpowiedni Content-type na:

- image/png
- image/gif
- image/jpeg

PHP – dynamiczna grafika

Następnie należy utworzyć nowy, pusty obrazek za pomocą funkcji **imagecreate()** - zwraca ona uchwyt do obrazka, coś na wzór uchwytu do pliku funkcji fopen().

Jako parametry tej funkcji trzeba podać wysokość i szerokość obrazka.

```
$img=imagecreate(200,200);
```

Można też utworzyć nowy obrazek na podstawie wzorca za pomocą funkcji **imagecreatefrompng()** .

```
$img=imagecreatefrompng("buttony/podklad1.png");
```

Jest to bardzo dobra metoda do tworzenia np. przycisków na stronie, gdyż wystarczy utworzyć jeden obrazek z tłem i później stworzyć tylko skrypt wstawiający tekst z parametru GET na gotowe tło.

PHP – dynamiczna grafika

- Po zakończeniu tworzenia obrazka trzeba go wysłać do przeglądarki.
- Służą do tego funkcje **imagepng()**, **imagejpeg()**, **imagegif()** i **imagewbmp()** .
- Pobierają one dwa argumenty:
  - jako pierwszy należy podać wskaźnik do obrazka,
  - drugi (opcjonalny) to nazwa pliku (używana przy zapisie obrazka na serwerze).
- W nazwach kilku funkcji obsługi grafiki występuje również format pliku, np. imagecreatefrompng czy imagepng - co oznacza to, że funkcja ta służy tylko do obsługi danego formatu pliku.

PHP – deklaracja kolorów

- Do deklarowania koloru służy funkcja **imagecolorallocate()** , która pobiera cztery argumenty.
- Należy podać kolejno zmienną-obiekt z obrazkiem oraz wartości składowe: czerwoną, zieloną i niebieską wybranego koloru (od 0 do 225).
- Funkcja zwraca zmienną-identyfikator koloru.

```
<?
$img=imagecreate(200,200);
$orange=imagecolorallocate($img,220,210,60);
$white=imagecolorallocate($img,225,225,225);
$black=imagecolorallocate($img,0,0,0);
?>
```

PHP – wypełnienie powierzchni

 Do wypełniania zwartych powierzchni służy funkcja **imagefill()** - należy podać cztery argumenty, zmienną id obrazka, współrzędną początkową x (left), współrzędną początkową y (top) oraz id koloru.

```
<?
imagefill($img,0,0,$black);
?>
```

PHP – linia

- Aby narysować linię, należy użyć funkcji **imageline()**.
- Argumenty, które należy podać, to imageline(*obrazek*, *x_poczatkowe*, *y_poczatkowe*, *x_koncowe*, *y_koncowe*, *kolor*).

```
<?
imageline($img,120,120,190,190,$white);
?>
```

PHP – prostokąt

 Aby narysować prostokąt, należy użyć funkcji **imagerectangle()**. Argumenty funkcji, to o*brazek*, *x_poczatkowe*, *y_poczatkowe*, *x_koncowe*, *y_koncowe*, *kolor*, przy czym:

- *x_początkowe* i *y_poczatkowe* to współrzędne lewego górnego wierzchołka kwadratu,
- *x_koncowe* i *y_koncowe* to współrzedne prawego dolnego wierzchołka.

 Istnieje też funkcja **imagefilledrectangle()** (te same argumenty), rysująca wypełniony kwadrat.

```
<?
imagerectangle($img,10,10,30,30,$orange);
imagefilledrectangle($img,30,30,50,50,$orange);
?>
```