Język PHP

Na podstawie podręcznika „Kwalifikacje E14”

S2.nasza-szkola.pl/~imie+1 literka nazwiska

Host: s2.nasza-szkola.pl
Login: imie+1 literka nazwiska
Hasło: 8lo……
Port: 4822

Formularze przypomnienie

Dokument

Wprowadzenie

PHP (ang. Hypertext Preprocesor) to skryptowy język programowania najczęściej wykorzystywany do tworzenia aplikacji internetowych. Skrypty pisane w PHP są umieszczane w kodzie HTML i wykonywane po stronie serwera. Klient otrzymuje tylko wynik wykonania skryptu.

PHP jest rozpowszechniany na zasadach **open source**.

Kod zapisany w skryptach PHP jest przetwarzany za pomocą silnika **Zend**. Nadzoruje on wykonywanie wszystkich operacji dotyczących przetwarzania kodu. Do działania niezbędny jest serwer WWW. Najczęściej jest nim serwer **Apache** lub **IIS**.

Serwer Apache

Najpopularniejszy i bezpłatny serwer WWW dostępny dla wielu systemów operacyjnych (Linux, Windows, Mac Os). Razem z interpreterem języka skryptowego PHP i bazą danych MySQL stanowi jedno z najpopularniejszych środowisk – tak zwaną platformę AMP.

Cechy serwera Apache

* Wielowątkowość
* Skalowalność
* Bezpieczeństwo
* Kontrola dostępu
* Uruchamianie witryn internetowych generowanych dynamicznie
* Niezależność od oprogramowania i sprzętu
* Dostępność jako open source

Serwer IIS

IIS (ang. Internet Information Services) jest komercyjnym serwerem WWW dostępnym dla systemów Windows Server i Windows. Wraz z serwerem baz danych MS SQL i środowiskiem programistycznym Visual Web Developer tworzy platformę WEB.PI.

Jest to technologia firmy Microsoft służąca do tworzenia dynamicznych stron WWW wykonywanych po stronie serwera.

Cechy serwera IIS

* Serwer oraz dostępne na nim witryny działają oddzielnie
* Witryny internetowe są izolowane i działają niezależnie od IIS oraz od siebie nawzajem

Struktura języka PHP

Skrypty PHP są plikami tekstowymi, do ich pisania można wykorzystywać dowolny edytor tekstu. Aby pliki były prawidłowo rozpoznawane przez serwer WWW muszą mieć rozszerzenie *.php*.

Umieszczając skrypt PHP w dokumencie HTML, należy zaznaczyć w kodzie gdzie zaczynają się i kończą polecenia PHP.

Rodzaj znacznika	Znacznik otwierający	Znacznik zamykający
Standardowy	<?php	?>
Skrócony	<?	?>
ASP	<%	%>
Skryptowy	<script language=„php”>	</script>

Przykłady

```
<?
echo „ Mój pierwszy skrypt PHP”;
?>

<?php
echo „ Mój pierwszy skrypt PHP”;
?>

<%
echo „ Mój pierwszy skrypt PHP”;
%>

<script language=„php”>
echo „ Mój pierwszy skrypt PHP”;
</script>
```

Instrukcja echo służy do wyświetlania danych.
Może być również stosowana z użyciem nawiasów np.

```
<?
echo( „ Mój pierwszy skrypt PHP”);
?>
```

Podobnie działa instrukcja print().

```
<?
print( „ Mój pierwszy skrypt PHP”);
?>
```

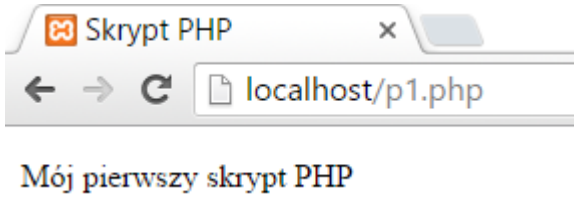
Instrukcje języka PHP muszą kończyć się średnikiem.

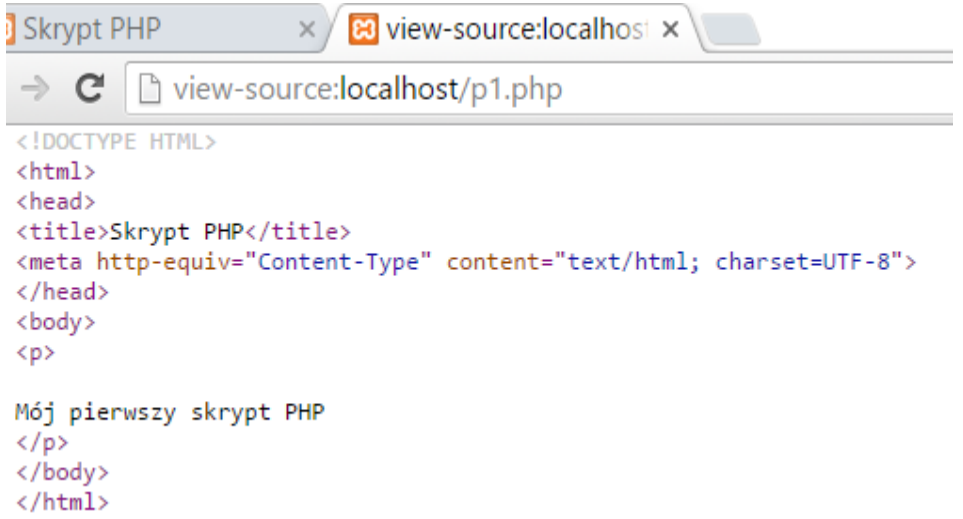
Blok PHP w kodzie HTML

```
<!DOCTYPE HTML>
<html>
<head>
<title>Skrypt PHP</title>
<meta charset="utf-8">
</head>
<body>
<p>

<?php
echo "Mój pierwszy skrypt PHP";
?>

</p>
</body>
</html>
```





Składnia języka PHP

Podstawową zasadą podczas tworzenia skryptów PHP jest oddzielanie kolejnych instrukcji znakiem średnika. Instrukcje mogą być pisane pojedynczo w jednym wierszu lub w kilku.

Występują trzy rodzaje komentarzy

* Komentarz blokowy /*……*/
* Komentarz jednowierszowy //
* Komentarz jednowierszowy uniksowy #

Składnia języka PHP. Zmienne.

Nazwa może być dowolna, ale musi spełniać następujące warunki:

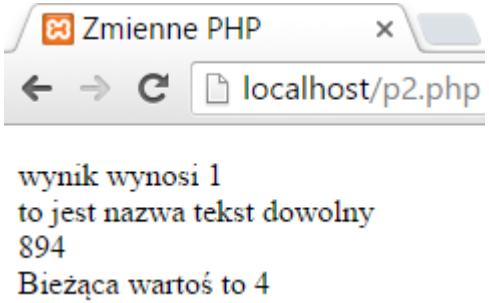
* musi zaczynać się od litery lub znaku podkreślenia
* może składać się jedynie z liter, cyfr i znaku podkreślenia
* w nazwach rozróżniane są małe i duże litery
* w nazwach można stosować polskie znaki

Przed nazwą zmiennej należy umieścić **znak $**

W języku PHP zmienna jest tworzona przy pierwszym jej użyciu, nie trzeba wcześniej deklarować zmiennej ani określać jej typu.

Zmienne. Przykład.

```
<?php
$x=1;
$nazwa_1="tekst dowolny";
$liczba7=894;
$ilosc=4;
echo("wynik wynosi $x<br/>");
echo("to jest nazwa $nazwa_1<br/>");
echo("$liczba7<br/>");
echo("Bieżąca wartość to $ilosc<br/>");
?>
```



Zmienne predefiniowane

Zmienne deklarowane wewnątrz skryptu lub funkcji są zmiennymi lokalnymi i istnieją tylko w obrębie skryptu lub funkcji. Do zmiennych lokalnych nie można odwoływać się w innych skryptach.

W PHP istnieje grupa zmiennych predefiniowanych, tzw. superglobalnych. Przechowują one informacje dotyczące konfiguracji bieżącego skryptu i najczęściej mają postać tablicy.

Zmienne predefiniowane

* **$_GET** – jest to tablica zawierająca zmienne przesyłane do skryptu za pomocą metody GET
* **$_POST** – jest to tablica zawierająca zmienne przesyłane do skryptu za pomocą metody POST
* **$_COOKIE** – jest to tablica zawierające zmienne przekazane z serwera do skryptu za pomocą cookies
* **$_FILES** – jest to tablica zawierająca zmienne przekazane do skryptu podczas przesyłania plików do serwera
* **$_SERVER** - jest to tablica zawierająca zmienne przekazane do skryptu przez serwer WWW. Są to dane takie jak wersja serwera, ścieżka do pliku, adres skryptu
* **$_ENV-** jest to tablica zawierająca wartości zmiennych środowiskowych serwera
* **$_REQUEST** – jest to tablica zawierająca zmienne przekazane do skryptu przez użytkownika. Zawiera dane z $_GET, $_POST, $_COOKIE oraz $_FILES
* **$_SESSION** – jest to tablica zawierająca zmienne zarejestrowane w bieżącej sesji
* **$GLOBALS** – jest to tablica zawierająca odniesienie do każdej zmiennej utworzonej przez użytkownika, która ma zasięg globalny dla danego skryptu.

Typy danych. Skalarne. Złożone. Specjalne.

Typy skalarne, czyli typy proste

* typ boolean
* typ integer
* typ float lub double
* typ string

Typy złożone

* typ array (tablicowy)
* typ object (obiektowy)

Typy specjalne

* typ resource
* typ null

Skalarne czyli proste

Typ boolean

Typ logiczny. Przyjmuje jedną z dwóch wartości prawda lub fałsz (true, false)

Typ integer

Typ liczb całkowitych

675 – dodatnia liczba całkowita

034 – dodatnia liczba całkowita zapisana w formacie ósemkowym

-021 – ujemna liczba całkowita zapisana w formacie ósemkowym

0xFF - dodatnia liczba całkowita zapisana w formacie szesnastkowym

-0x0C – ujemna liczba całkowita zapisana w formacie szesnastkowym

Typ float

Liczba zmiennoprzecinkowa. Może przedstawiać zarówno dodatnie, jak i ujemne liczby rzeczywiste. Można używać notacji wykładniczej.

1,47; -2,346;0.17E2;-1.0E-2

Typ string

Typ łańcuchowy, który służy do zapamiętywania ciągu znaków

Typ string

Łańcuch znaków można utworzyć, korzystając z jednego z czterech sposobów:

* używając znaków apostrofu,
* używając znaków cydzysłowu,
* używając składni heredoc,
* używając składni nowdoc.

Typ string - znaki apostrofu

Ciąg znakowy można utworzyć poprzez ujęcie go w znaki apostrofu. Jeżeli apostrof zostanie użyty jako jeden ze znaków ciągu, to należy poprzedzić go znakiem \.

```
<?php
echo 'To jest symbol apostrofu \' użyty w kodzie PHP';
?>
```

Lub

```
<?php
$z= 'to jest tekst';
echo  $z;
?>
```

Typ string – znaki cudzysłowu

```
<?php
$z= ”to jest tekst”;
echo  $z;
?>
```

Typ string – składnia heredoc

* Jeżeli jest używana składnia *heredoc*, ciąg znaków trzeba rozpocząć od sekwencji <<<. Po tej sekwencji musi wystąpić identyfikator. Ten sam identyfikator musi zostać użyty na końcu ciągu znakowego. Linia zawierająca identyfikator kończący ciąg nie może zawierać innych znaków oprócz tego identyfikatora i średnika.

```
<?php
$napis=„napis”;
$tekst=<<<ABC
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Maecenas porttitor congue massa. Fusce posuere, magna sed pulvinar ultricies, purus lectus malesuada libero, sit amet commodo magna eros quis urna.
i jakiś $napis
ABC;
echo $tekst;
?>
```

* **Ten sposób tworzenia napisów jest wykorzystywany, kiedy są one długie i składają się z wielu linii.**

Typ string – składnia *nowdoc*

Składnia *nowdoc* jest podobna do składni *heredoc*. Różnica polega na umieszczeniu identyfikatora w znakach apostrofu.

Różnica w działaniu polega na tym, że gdy w ciągu znaków wystąpi odwołanie do zmiennej, nie jest ona zamieniana na odpowiadającą jej wartość.

```
<?php
$napis=„napis”;
$tekst=<<<’ABC’
Lorem ipsum dolor sit amet, consectetuer adipiscing elit ultricies, purus lectus malesuada libero, sit amet commodo magna eros quis urna. i jakiś $napis
ABC;
echo $tekst;
?>
```

Typ array

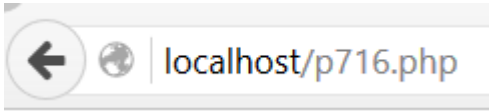
Struktury tablicowe w PHP są tworzone na bieżąco przez przypisanie wartości dla poszczególnych indeksów albo za pomocą instrukcji *array()*. W PHP można tworzyć tablice zwykle, indeksowane kolejnymi liczbami całkowitymi (**tablice indeksowane**),

oraz **tablice asocjacyjne**, które indeksowane są kluczem.

Tablice indeksowane

Polecenie *array* dla tablic indeksowanych tworzonych automatycznie

$tab= array(t1,t2,t3,……tn);



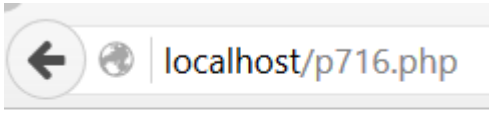
napis2

Składnia tworzenia tablicy indeksowanej przez ręczne przypisanie indeksu

$tab[0]=t1;
$tab[1]=t2;
$tab[2]=t3;
.
.
$tab[n]=tn;

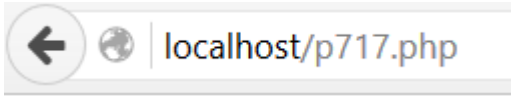
Przykłady tablice indeksowane

```php
<?php
$ks=array("napis1","napis2","napis3");
echo $ks[1];
?>
```



napis2

```php
<?php
$im[0]="adam";
$im[1]="jan";
$im[2]="ewa";
$im[3]="bartek";
echo $im[2];
?>
```

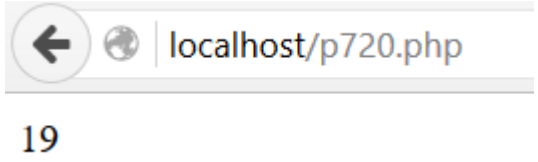


ewa

Tablice asocjacyjne

Inaczej zwane tablicami skojarzeniowymi(ang. associative array) to tablice, w których zamiast indeksów liczbowych używa się identyfikatorów (tzw. Kluczy). W danych tego typu przechowywane są pary danych; unikatowy klucz i wartość.

```
<?php
$osoba["nazwisko"]="kowalski";
$osoba["imie"]="jan";
$osoba["wiek"]=19;
echo $osoba["wiek"];
?>
```



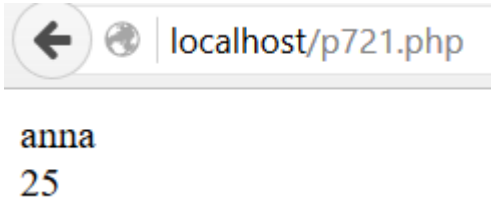
Można inaczej:

```
$osoba=array(„nazwisko”=>”kowalski”,”imie”=>”jan”,”wiek=>19);
```

Tablice wielowymiarowe

Inaczej tablice z tablic

```
<?php
$dane=array(
        array("nazwisko"=>"kowalski", "imie"=>"jan","wiek"=>19),
        array("nazwisko"=>"lis", "imie"=>"bartek","wiek"=>29),
        array("nazwisko"=>"malinowska", "imie"=>"anna","wiek"=>25),
);
echo $dane[2]["imie"];
echo "<br>";
echo $dane[2]["wiek"];
```



Inne typy zmiennych

* Typ object - służący do przechowywania obiektów
* Typ resource – typ specjalny wskazujący, że zmienna przechowuje odwołanie do zasobu zewnętrznego, utworzonego za pomocą specjalnych funkcji.
* Typ null – przeznaczony dla zmiennych, które zostały zadeklarowane, ale nie przypisano im żadnych wartości

Zmiana typu zmiennej (*cast*)

W języku PHP podczas przypisywania zmiennej nowej wartości ponownie ustalany jest jej typ. Można jednak zmienić typ danych jeśli zachodzi taka potrzeba. Można tego dokonać za pomocą rzutowania (*cast*) jednorazowo, lub trwale za pomocą funkcji settype().

Rzutowanie typów odbywa się przez podanie nowego typu w nawiasie przed zmienną lub wartością, której typ należy zmienić.

integer – rzutowanie do typu całkowitego,

float(double) -rzutowanie do typu rzeczywistego,

string – rzutowanie do ciągu tekstowego,

array – rzutowanie do tablicy,

object – rzutowanie do obiektu.

```
<?php
$x=76.98;
$y=(integer)$x;
echo "$x <br/>";
echo "$y<br/>";
?>
```

Zmiana typu zmiennej (*settype()*)

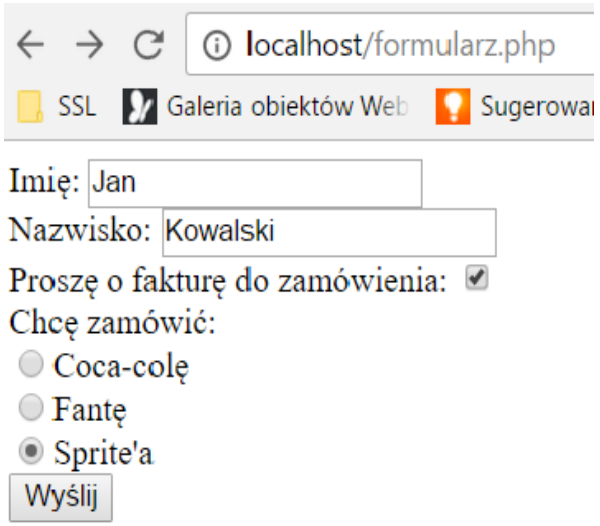
Zmianę typu w sposób trwały przeprowadza się za pomocą funkcji *settype()*. Funkcja ta posiada dwa argumenty. Pierwszy jest nazwą zmiennej, której nadajemy nowy typ, drugim parametr określający nowy typ zmiennej. Parametr ten może przyjmować wartości :*integer, float, string, array, object*.

settype($zmienna, ,nowy_typ');

```
<?php
$x=76.98;
echo " wartość zmiennej x :$x<br/>";
settype($x,'string');
echo "Wartość zmiennej(string) x: $x<br/>";
settype($x,'integer');
echo "Wartość zmiennej(integer) x: $x<br/>";
?>
```

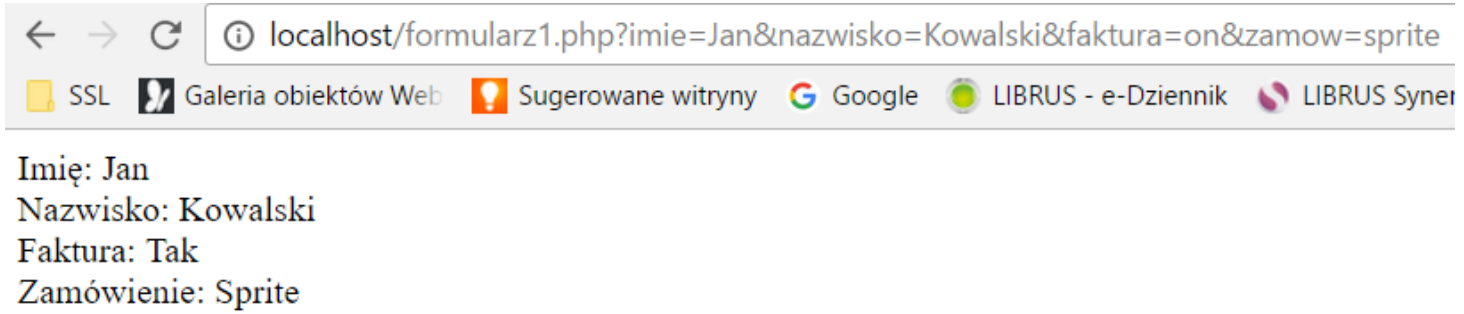
Przekazywanie danych z formularza

```
<html> <head>  <title>
  Test formularza
  <meta http-equiv="Content-Type" content="text/html";charset =UTF-8>
 </title> </head>
<body>
<form action="formularz1.php" method="GET">
 Imię: <input type=text name="imie"/><br/>
 Nazwisko: <input type=text name="nazwisko"/><br/>
 Proszę o fakturę do zamówienia: <input type=checkbox
name="faktura"/><br/>
 Chcę zamówić:<br/>
 <input type=radio name=zamow value="kola"/>Coca-colę<br>
 <input type=radio name=zamow value="fanta"/>Fantę<br>
 <input type=radio name=zamow value="sprite"/>Sprite'a<br>
 <input type=submit value="Wyślij"/>
 </form>
</body>
</html>
```



Odbieranie danych

```
<html> <head>  <title>
  Zamówienie
  <meta http-equiv="Content-Type" content="text/html";charset =UTF-8>
 </title> </head>
<body>
 Imię: <?php echo $_GET['imie']?><br>
 Nazwisko: <?php echo $_GET['nazwisko']?><br>
 Faktura: <?php echo ($_GET['faktura'] == 'on' ? 'Tak' : 'Nie')?><br>
 Zamówienie: <?php switch($_GET['zamow']){
  case "kola":
    echo "Coca-cola";
    break;
  case "fanta":
    echo "Fanta";
    break;
  case "sprite":
    echo "Sprite";
    break;
  }
?>
 </body></html>
```



Zadania

* Utwórz trzy zmienne. Pierwszej przypisz wartość liczby całkowitej, drugiej wartość liczby rzeczywistej, trzeciej ciąg znaków. Utwórz w języku PHP skrypt, który będzie wyświetlał wartości tych zmiennych na ekranie. Przetestuj zmianę typu zmiennych.

Operatory i wyrażenia

Operatory służą do wykonywania działań na zmiennych. W języku PHP można je podzielić na operatory:

Arytmetyczne

* porównania
* bitowe
* logiczne
* przypisania
* inkrementacji i dekrementacji
* pozostałe

Operatory arytmetyczne

Służą do wykonywania operacji arytmetycznych

Operator	Działanie	Przykład
+	Dodawanie	$a + $b
-	Odejmowanie	$a - $b
*	Mnożenie	$a * $b
/	Dzielenie	$a / $b
%	Dzielenie modulo	$a % $b

Operatory porównania

Operatory porównania porównują argumenty; ich wynikiem jest wartość logiczna *true* lub *false*.

Operator	Działanie	Przykład
==	Wynik *true*, gdy argumenty są równe.	$a == $b
!=	Wynik *true*, gdy argumenty są różne.	$a != $b
===	Wynik true, gdy argumenty są tego samego typu i są równe.	$a === $b
!==	Wynik true, gdy argument są różne lub są różnych typów.	$a !== $b
>	Wynik true, gdy pierwszy argument jest większy od drugiego	$a > $b
<	Wynik true, gdy pierwszy argument jest mniejszy od drugiego	$a < $b
>=	Wynik true, gdy pierwszy argument jest większy od drugiego, lub mu równy.	$a >= $b
<=	Wynik true, gdy pierwszy argument jest mniejszy od drugiego, lub mu równy.	$a <= $b
<>	Wynik true, gdy argumenty są różne.	$a <> $b

Operatory bitowe

Operatory bitowe umożliwiają wykonanie operacji na poszczególnych bitach liczb.

Operator	Działanie	Przykład
&	Iloczyn bitowy (AND)	$a & $b
|	Suma bitowa (OR)	$a | $b
~	Negacja bitowa (NOT)	$a ~ $b
^	Bitowa różnica arytmetyczna	$a ^ $b
>>	Przesunięcie bitowe w prawo	$a >> n
<<	Przesunięcie bitowe w lewo	$a << n

Operatory logiczne

Operatory logiczne wykonują operacje na argumentach, które posiadają wartość logiczną (*true* lub *false*).

Operator	Działanie	Przykład
and	Iloczyn logiczny	$a and $b
&&	Iloczyn logiczny	$a && $b
or	Suma logiczna	$a or $b
||	Suma logiczna	$a || $b
!	Negacja logiczna (NOT)	! $a
xor	Różnica symetryczna	$a xor $b

* Negacja logiczna zmienia wartość argumentu na przeciwną.
* Wynik różnicy symetrycznej przyjmuje wartość *true* tylko wtedy, gdy obydwa argumenty mają różną wartość. Gdy obydwa argumenty mają równą wartość wynikiem jest *false*.

Operatory przypisania

Operatory przypisania przypisują wartość argumentom znajdującym się po lewej stronie operatora.

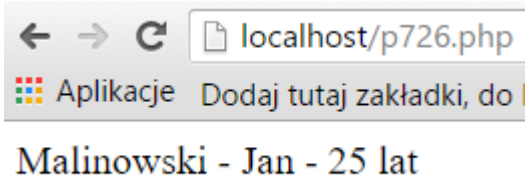
Operator	Przykład	Znaczenie
=	$x = 10	$x = 10
+=	$ + = 5	$x = $x + 5
-=	$x - = 5	$x = $x - 5
*=	$x * = 5	$x = $x * 5
/=	$x / = 5	$x = $x / 5
%=	$x % = 5	$x = $x % 5
.=	$x . = „ab”	$x = $x . „ab”

Operator konkatenacji

Operator konkatenacji służy do łączenia ciągów znakowych. Niezależnie od typów danych, które będą łączone za pomocą tego operatora, są one traktowane jako ciągi znaków i rezultat ich łączenia zawsze jest ciągiem znaków.

Operator	Działanie	Przykład
.	Łączenie łańcuchów znakowych	„łańcuch” . „tekstowy”

```
<?php
$osoba["nazwisko"]="Malinowski";
$osoba["imie"]="Jan";
$osoba["wiek"]=25;
echo $osoba["nazwisko"]." - ".$osoba["imie"]. " - ".
$osoba["wiek"]." lat";
?>
```



Stałe

W języku PHP występują stałe, czyli identyfikatory, których wartości nie ulegają zmianie. Nazwy stałych w odróżnieniu od zmiennych nie posiadają znaku $.

Zasięg stałych jest globalny i można odwoływać się do nich w każdym miejscu skryptu. Do ich definiowania służy funkcja *define()*, która posiada dwa argumenty: nazwę stałej i przypisaną do niej wartość (boolean, integer, float, string).

define(„NAZWA_STALEJ”,wartość);

np.:

```
define(„WIEK”,”21”);
echo „Mamy wiek” . WIEK;
```

Instrukcje sterujące instrukcja warunkowa

Instrukcja warunkowa może występować w kilku wersjach:

```
if(warunek){
Instrukcja1;
Instrukcja2;
Instrukcja3;
}
```

```
if(warunek)
Instrukcja1;
else
Instrukcja2;
```

```
if(warunek1)
Instrukcja1;
else if (warunek2)
Instrukcja2;
else if (warunek_n)
Instrukcja n;
else
Instrukcja n+1;
}
```

Instrukcje sterujące instrukcja switch

Jest to instrukcja wyboru. Wyrażenie występujące w tej konstrukcji jest porównywane z listą szukanych wartości aż do znalezienia pasującej wartości.

```
switch (wyrażenie){
case wartość1:
instrukcje1;
break;
case wartość2:
instrukcje2;
break;
case wartość1:
instrukcje1;
break;
default:
instrukcje4;
}
```

```
<?php
$kolor="red";
switch($kolor)
{
case "red":
        echo "kolor czerwony";
        break;
case "blue":
        echo "kolor niebieski";
        break;
case "green":
        echo "kolor zielony";
        break;
default:
        echo "brak koloru";}
?>
```

Instrukcje sterujące operator warunkowy

Operator warunkowy działa podobne jak instrukcja if. Zwraca wartość jednego z dwóch wyrażeń rozdzielonych dwukropkiem.

warunek ? wartość1 : wartość2;

```
<?php
$x=15;
$wynik = ($x<0)?"liczbaujemna":"liczba
dodatnia";
echo "Wartość zmiennej x jest $wynik";
?>
```

Pętle

W języku PHP występują następujące rodzaje pętli;

for, while, do…..while, foreach.

```
for ($i=1; $i<5;$i++){
echo ,,pętla wykonana $i razy”;
}
```

```
$i=0;
while ($i++<5){
echo ,,pętla wykonana $i razy”;
}
```

```
$i=1;
do{
echo ,,Pętla wykonana $i razy”;
while($i++ <5);
}
```

Pętla *foreach*

Pętla foreach umożliwia dostęp do elementów tablicy lub właściwości obiektu. Może występować w postaci:

```
foreach($tablica as $wartość){
instrukcje;
}
```

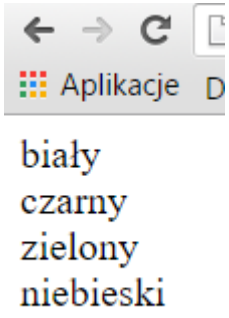
Lub

```
foreach($tablica as $klucz=>$wartość){
instrukcje;
}
```

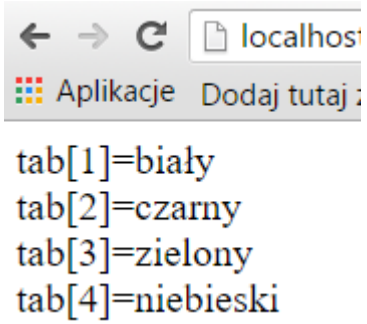
W drugim przypadku oprócz wartości elementu tablicy otrzymujemy wartość aktualnego klucza.

Zastosowanie

```
<?php
$tab=array(
        1=>'biały',
        2=>'czarny',
        3=>'zielony',
        4=>'niebieski'
);
foreach($tab as $x){
        echo "$x <br/>";
};
?>
```




```
<?php
$tab=array(
        1=>'biały',
        2=>'czarny',
        3=>'zielony',
        4=>'niebieski'
);
foreach($tab as $kl=>$x){
        echo "tab[$kl]=$x <br/>";
};
?>
```



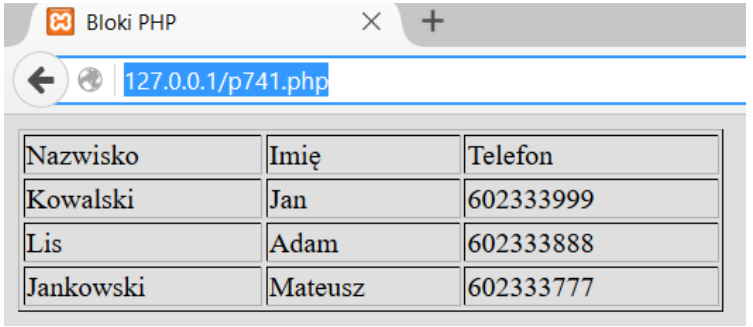
Ćwiczenia

* Napisz skrypt, który będzie wyświetlał liczby nieparzyste z przedziału od 0 do 30. Liczby powinny być wyświetlane w kolumnie od wartości największej do najmniejszej.
* Napisz skrypt, który umieści w tabeli oceny semestralne z pięciu przedmiotów, a następnie wyświetli dane z tabeli w postaci dwóch kolumn. Pierwsza kolumna będzie zawierać nazwy przedmiotów, a druga oceny z tych przedmiotów.

Naprzemienne bloki kodu PHP i HTML

Jeżeli wewnątrz bloku PHP wystąpi kod HTML, który będzie wykonywany opcjonalnie – tylko wtedy, gdy zostanie spełniony określony warunek – to można zdefiniować przełączanie się do trybu HTML wewnątrz bloku warunkowego PHP.

```
<!DOCTYPE HTML>
<html>
<head>
<title>Bloki PHP</title>
<meta http-equiv="Content-Type" content="text/html";charset =UTF-8>
</head>
<body bgcolor=#dfdfdf>
        <?php
        $tx=true;
        if ($tx){
        ?>
        <table align="left" border="1" width="400" hspace="40" vspace="20" cell-spcing="4">
                <tr><td>Nazwisko</td><td>Imię</td><td>Telefon</td></tr>
                <tr><td>Kowalski</td><td>Jan</td><td>602333999</td></tr>
                <tr><td>Lis</td><td>Adam</td><td>602333888</td></tr>
                <tr><td>Jankowski</td><td>Mateusz</td><td>602333777</td></tr>
                </table>
        <?php
        }
        ?>
</body>
</html>
```



Ćwiczenie

Połącz skrypt PHP z poprzedniego slajdu z kodem HTML i dodaj w pierwszej kolumnie Lp. z zastosowaniem pętli w PHP.

Funkcje

Funkcja to ciąg instrukcji stanowiący blok kodu, który może być wielokrotnie wykorzystywany w różnych programach lub jego miejscach.

Istnieją dwa rodzaje funkcji. Funkcje, które zostały wbudowane w język, oraz funkcje zdefiniowane przez programistę.

Definiowanie funkcji

Funkcja bezparametrowa

```
<?php
function napis() {
echo "<h2>Program w PHP";
}
napis();
?>
```

Funkcja z parametrami

```
<?php
function dodaj($a,$b) {
$c=$a+$b;
echo "Wynik dadawania $a + $b wynosi $c";
}
dodaj(15,7);
?>
```

Zwracanie wartości przez funkcje i zasięg zmiennych

Jeżeli chcemy, aby funkcja zwracała wartość, która będzie wykorzystywana w innych działaniach, należy użyć słowa kluczowego *return*.

```
<?php
function dodaj($a,$b) {$c=$a+$b;
return $c;}
$x=15;//zmienna globalna
$y=7;
$suma=dodaj($x,$y);
echo "Wynik dadawania $x + $y wynosi $suma";
?>
```

```
<?php
$zm=1;
function pokaz(){
global $zm;
echo "Wartość zmiennej \$zm
wynosi $zm";}
pokaz ();
?>
```

Instrukcja *global*

global $nazwa_zmiennej;

Lub odwołanie do tablicy

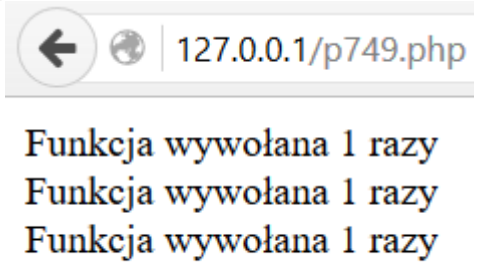
$GLOBALS[,nazwa_zmiennej'];

```
<?php
$zm=7;
function pokaz(){   global $zm;
echo "Wartość zmiennej \$zm wynosi:";
echo $GLOBALS['zm'];}
pokaz ();
?>
```

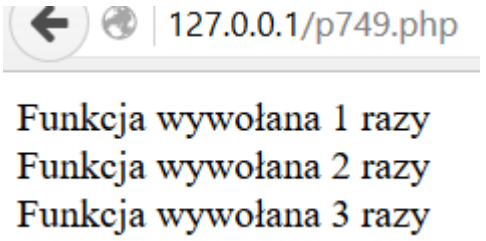
Zmienne statyczne

* To zmienne lokalne funkcji, które zachowują swoją wartość między wywołaniami funkcji.

```
<?php
function funk(){
$i=1;
echo "Funkcja wywołana $i razy<br>";
$i++;
}
funk();
funk();
funk();
?>
```




```
<?php
function funk(){
static $i=1;
echo "Funkcja wywołana $i razy<br>";
$i++;
}
funk();
funk();
funk();
?>
```



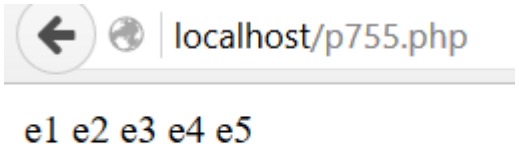
Funkcje wbudowane

W języku PHP Istnieje duża grupa predefiniowanych funkcji, które są przydatne podczas tworzenia aplikacji WWW. Funkcje te zostały podzielone na grupy tematyczne – ułatwia to znalezienie funkcji wykonującej określone zadania.

Funkcje tablic

Do najpopularniejszych należą *count()* i *sizeof()*, które zliczają liczbę elementów w tablicy.

```
<?php
$tab = array("e1", "e2", "e3", "e4",
"e5");
$dl = count($tab);
for($i =0; $i <$dl; $i++) {
echo $tab[$i], " ";
}
?>
```



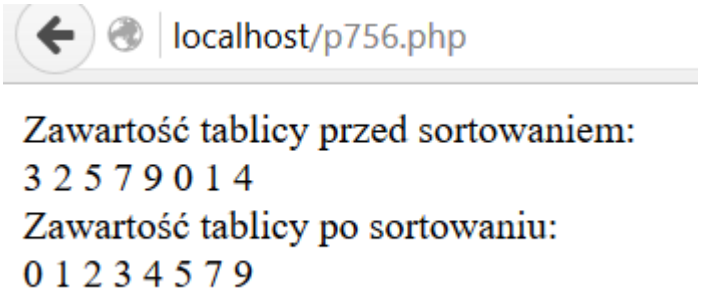
Funkcje sortowania

Sortowanie polega na uprządkowaniu elementów tablicy w określonym porządku.

* *sort()* – sortuje elementy tablicy indeksowanej w kolejności od najmniejszego do największego
* *rsort()* - sortuje elementy tablicy indeksowanej w kolejności od największego do najmniejszego
* *asort()* – sortuje elementy tablicy asocjacyjnej według zawartości, w kolejności od najmniejszego do największego
* *arsort()* - sortuje elementy tablicy asocjacyjnej według zawartości, w kolejności od największego do najmniejszego
* *ksort()* - sortuje elementy tablicy asocjacyjnej według klucza, w kolejności od najmniejszego do największego
* *krsort()* - sortuje elementy tablicy asocjacyjnej według klucza, w kolejności od największego do najmniejszego

Przykład sortowania

```
<?php
$tab = array(3, 2, 5, 7, 9, 0, 1, 4) ;
echo "Zawartość tablicy przed sortowaniem: <br/>";
foreach($tab as $x) {
echo "$x ";
}
sort($tab);
echo "<br/>Zawartość tablicy po sortowaniu: <br/>";
foreach($tab as $x) {
echo "$x ";
}
?>
```



Funkcje wyszukiwania

array_search() – funkcja posiada dwa argumenty. Pierwszym argumentem funkcji jest poszukiwana wartość, drugim przeszukiwana tablica. Jeżeli szukana wartość zostanie znaleziona, funkcja zwróci wartość *true*, w przeciwnym razie zwróci wartość *false*.

Zadanie

Napisz skrypt wyszukujący podaną liczbę przez użytkownika,

Rozbuduj skrypt na potrzeby LOTTO, czyli w tablicy masz podane 6 liczb z zakresu od 1 do 49, użytkownik wpisuje swoje 6 liczb z kuponu i dostaje odpowiedź np. „masz 6 trafień, zostałeś milionerem”, albo „przykro mi, grasz dalej”

Podpowiedź

```
<body>
<h1>Podaj szukane liczby</h1>
        <form action="lotto.php" method="post">
        Pierwsza liczba:
        <input type="text" name="l1" />
        <br /><br />
        <input type="submit" value="Wyślij zamówienie" />
        </form>
</body>
```

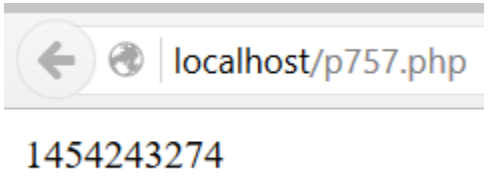
```
<?php
        // wylosowane liczby w lotto
        $tab= array(1,2,3,4,5,6);
        $l1 = $_POST['l1'];
        echo "szukana liczba";
        echo $l1;
        $z1=array_search($l1,$tab);
        if($z1!=0){
                echo "<br/>Jest taka liczba";}
        else
                {echo "<br/>nie ma takiej liczby";}
?>
```

Funkcje daty i czasu

Funkcja *time()*

Zwraca informację na temat bieżącej daty i czasu. Nie posiada żadnych argumentów. Informacje na temat bieżącej daty i czasu są zwracane w postaci liczby. Liczba ta odpowiada liczbie sekund, które upłynęły od godziny 00:00:00 1 stycznia 1970 roku.

```
<?php
echo time();
?>
```



Funkcje daty i czasu

Funkcja *getdate()*

Ma ona postać:

getdate([znacznik_czasu])

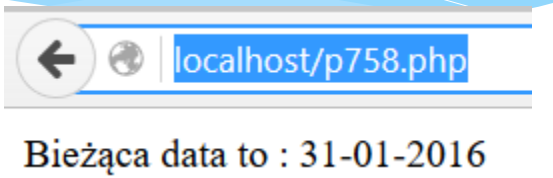
Argument znacznik_czasu jest opcjonalny. Jeżeli nie zostanie podany, działania wykonywane przez funkcję będą się odnosić do daty bieżącej. Wynikiem wykonywania funkcji jest tablica asocjacyjna zawierająca dane dotyczące daty i czasu. Indeksy oraz wartości tej tablicy dotyczą poszczególnych elementów wchodzących w skład daty i czasu.

Funkcje daty i czasu

Nazwa indeksu	Znaczenie indeksu	Opis
seconds	Liczba sekund	Od 0 do 59
minutes	Liczba minut	Od 0 do 59
hours	Godzina	Od 0 do 23
mday	Dzień miesiąca	Od 1 do 31
wday	Dzień tygodnia w postaci liczby	Od 0 (niedziela) do 6 (sobota)
mon	Miesiąc w postaci liczby	Od 1 do 12
year	Rok w postaci czterocyfrowej	Np.2016
yday	Numer kolejnego dnia roku	Od 0 do 365
weekday	Nazwa dnia tygodnia (po angielsku)	Np. Monday
month	Nazwa miesiąca (po angielsku)	Np. january
0	Aktualny znacznik czasu	

Przykład

```
<?php
$data=getdate();
$dzien=$data["mday"];
$miesiac=$data["mon"];
$rok=$data["year"];
if($dzien<10) $dzien="0".$dzien;
if($miesiac<10) $miesiac="0".$miesiac;
echo "Bieżąca data to : $dzien-$miesiac-$rok";
?>
```



Na podstawie ćwiczenia rozbuduj program, który wyświetli informację szczegółową w postaci

Bieżąca data to: niedziela 27 stycznia 2019

Funkcje daty i czasu

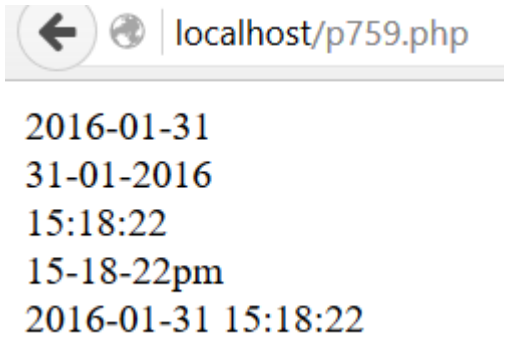
Funkcja date()

date(format[,znacznik_czasu])

Parametr format określa, jakie informacje na temat daty i czasu powinny zostać zwrócone. Parametr znacznik_czasu zawiera opisany wcześniej znacznik (slajd 55). Jest on opcjonalny.

Parametr format jest ciągiem znaków składającym się ze znaczników.

```
<?php
echo date("Y-m-d")."<br/>";
echo date("d-m-Y")."<br/>";
echo date("G:i:s")."<br/>";
echo date("H-i-sa")."<br/>";
echo date("Y-m-d G:i:s")."<br/>";
?>
```



Funkcje daty i czasu

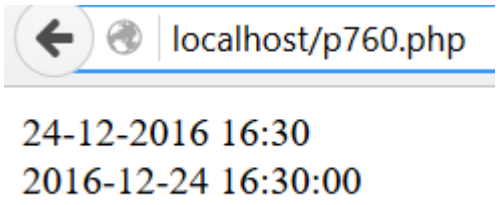
Znacznik	Znaczenie	Opis
a	am lub pm (przed lub popołudniu	am,pm
A	AM lub PM (przed lub popołudniu	AM, PM
c	Data i czas zgodny z formatem ISO8601	2016-01-01T15:09:07
d	Dzień miesiąca w formacie z zerem na początku	Od 01 do 31
D	Dzień miesiąca w formacie trzyliterowego skrótu	Mon, Tue
F	Pełna nazwa miesiąca	January
g	Godzina w formacie 12-godzinnym bez zera na początku	Od 1 do 12
G	Godzina w formacie 24-godzinnym bez zera na początku	Od 1 do 24
H	Godzina w formacie 24-godzinnym z zerem na początku	Od 01 do 24
i	Liczba minut z zerem na początku	Od 01 do 59
l	Nazwa dnia tygodnia	Monday
m	Miesiąc w postaci liczby z zerem na początku	Od 01 do 12
s	Liczba sekund z zerem na początku	Od 01 do 59
Y	Rok w postaci 4 znaków	2016

Funkcje daty i czasu

Funkcja mktime()

Zwraca znacznik czasu daty podanej jako argument funkcji. Może posiadać od 0 do 6 argumentów podanych w postaci liczb całkowitych. Są to kolejno

* Godzina
* Minuta
* Sekunda
* miesiąc
* Dzień
* Rok




```
<?php
$czas=mktime(16,30,0,12,24,2016);
echo date("d-m-Y G:i",$czas)."<br/>";
echo date("Y-m-d G:i:s",$czas);
?>
```

Znacznik czasu, który zwróci funkcja *mktime()*, może zostać wykorzystany w funcji *getdate()* i *date()*.

Ćwiczenia

* Napisz skrypt wyświetlający liczbę dni, które upłynęły od początku roku, oraz liczbę dni, które pozostały do końca roku.
* Napisz skrypt, który będzie wyświetlał dzień tygodnia w postaci ***Dzisiaj jest środa***
* Napisz skrypt, który będzie wyświetlał nazwy dni tygodnia tzw. „Długich weekendów” w roku 2016, 2017, 2018. (6 stycznia, 1, 3 maja, 15 sierpnia, 25,26 grudzień).

Pliki i katalogi

Dołączanie plików

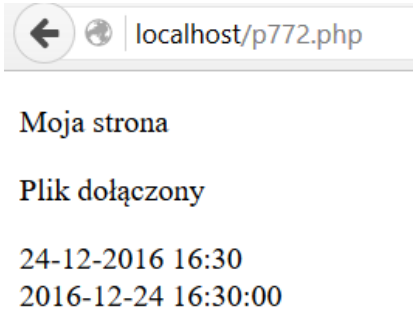
Jeżeli skrypt PHP zawiera dużą ilość kodu, to można go podzielić i zapisać w kilku oddzielnych plikach. Do ponownego ich połączenia można użyć instrukcji *include* lub *require.*

include(‚nazwa pliku’) lub include ‚nazwa_pliku’

require (‚nazwa_pliku’) lub require ‚nazwa_pliku’

```
<!DOCTYPE HTML>
<html>
        <head>
        <title>Dodączanie kodu</title>
        </head>
        <body>
                <p>Moja strona</p>
<?php
include 'p772a.php';
?>
</body>
</html>
```

```
<?php
echo "<p>Plik dołączony</p>";
$czas=mktime(16,30,0,12,24,2016);
echo date("d-m-Y G:i",$czas)."<br/>";
echo date("Y-m-d G:i:s",$czas);
?>
```



Operacje na plikach

Do ustalenia, czy plik istnieje, służy funkcja *file_exists()* zapisywana w postaci:

file_exists(‚nazwa_pliku’);

Np.:

```
if (file_exists(‚skrypt.php’)){
echo „Plik istnieje”;
}
```

Funkcja *is_file()* sprawdza czy podany jako argument ciąg jest plikiem.

is_file(‚plik’);

Np.:

```
$p=‚plik.txt’
if(is_file($p)){
echo „to jest plik”;
}
```

Do określenia rozmiaru pliku służy funkcja *filesize()* w postaci :

filesize(‚nazwa_pliku’);

Funkcja zwraca wartość typu integer, która określa wielkość pliku w bajtach.

Do tworzenia pliku używana jest funkcja *touch().*

touch(‚nazwa_pliku’);

Funkcja tworzy pusty plik o podanej nazwie.

Do usuwania istniejącego pliku służy polecenie *unlink()*

unlink (‚nazwa_pliku’);

Funkcja zwraca wartość prawdy, jeżeli plik został usunięty.

Za pomocą funkcji fopen() można otwierać istniejące pliki.

fopen(‚nazwa_pliku’,tryb_otwarcia);

* r -tylko do odczytu
* w- tylko do zapisu
* a – tryb dopisywania

$p=fopen(‚dane.txt’, ‚r’);

Plik zostanie otwarty tylko do odczytu. Funkcja zwraca wartość *false*, jeżeli plik nie może być otwarty.

$p=fopen(‚dane.txt’, ‚w’);

Po wykonaniu tego kodu znajdujące się dane pliku zostaną usunięte, a nowe nadpisane. Jeżeli plik nie istniał zostanie utworzony.

$p=fopen(‚dane.txt’, ‚a’);

Nowe dane zostaną dopisane na końcu pliku.

Do zamykania służy funkcja *fclose(deskryptor)*

Deskryptor to wartość zwrócona przez funkcję *fopen()*.

```
if($p=fopen(‚dane.txt’, ‚w’)){
// działanie… …
fclose($p);
}
```

Zapisujemy

Do zapisywania stosowana jest również funkcja *fwrite()* w postaci :

fwrite(feskryptor, ciąg_znaków)

Pierwszy argument oznacza plik zwrócony za pomocą funkcji *fopen()*, drugi to ciąg znaków, które mają być zapisane. Funkcja zwróci *false*, jeżeli zapisanie ciągu znaków do pliku się nie powiodło.

```
<?php
$tekst="Lorem ipsum dolor sit amet, consectetuer adipiscing elit.
Maecenas porttitor congue massa. Fusce posuere, magna sed pulvinar
ultricies, purus lectus malesuada libero, sit amet commodo magna eros
quis urna.";
if(!$p=fopen('dane.txt','a')){
        echo "nie można otworzyć pliku";
        }
        else{
                if(fwrite($p,$tekst)===false){
                        echo "Zapis do pliku nie powódł się";
                }
                else{
                echo "zapisałem".$tekst;
                } fclose($p);}
?>
```

Zadanie

* Wykorzystując zdobyte wiadomości z zakresu zapisywania tekstu do pliku, napisz program, który zapisze tekst wprowadzony na stronie w polu tekstowym do pliku „dane.txt”

Odczyt danych(fgets)

Po otwarciu pliku można odczytywać pojedyncze wiersze za pomocą funkcji *fgets()*.

fgets(deskryptor, ile_znaków)

Funkcja fgets() często stosowana jest razem z funkcją *feof()*.

feof(deskryptor)

```
<?php
if(!$p=fopen('dane.txt','r')){
        echo "nie można otworzyć pliku";
        }
        else{
                while(!feof($p)){
                        $w=fgets($p,100);
                        echo "$w<br/>";
                }
                fclose($p);
        }
?>
```

Odczyt danych(fgetc)

Do odczytywania pojedynczych znaków służy funkcja fgetc(deskryptor)

Funkcja zwraca ciąg zawierający 1 znak. Po wykonaniu funkcji wskaźnik pliku jest przesuwany o 1 znak do przodu. Gdy jest osiągnięty koniec pliku, funkcja zwraca wartość *false*.

```
<?php
if(!$p=fopen('dane.txt','r')){
        echo "nie można otworzyć pliku";
        }
        else{
                while(($z=fgetc($p))!==false){
                        echo "<br/>$z";
                }
                fclose($p);
        }
?>
```

Odczyt danych (fread)

Do odczytu bloków danych służy funkcja fread()

fread(feskryptor,ile_znaków)

W przykładzie dane odczytywane są w blokach 32-bajtowych i wysyłane do przeglądarki.

```
<?php
if(!$p=fopen('dane.txt','r')){
        echo "nie można otworzyć pliku";
        }
        else{
                while(!feof($p)){
                        $b=fread($p,32);
                        echo "$b<br/>";
                }
                fclose($p);
        }
?>
```

Odczyt danych

Funkcja readfile()

readfile(‚nazwa_pliku’)

Wysyła zawartość pliku podanego jako argument do przeglądarki.

Funkcja file_get_contents()

file_get_contents(‚nazwa_pliku’)

Zwraca odczytaną zawartość pliku podanego jako argument w postaci ciągu tekstowego.

Funkcja file()

file(‚nazwa_pliku’)

Funkcja odczytuje całą zawartość pliku i zwraca tę zawartość w postaci tablicy. Każda komórka tablicy zawiera kolejny wiersz odczytanego tekstu.

Ćwiczenie

Wykorzystując funkcje

* fgets()
* fgetc()
* fread()
* readfile()

wyświetl tekst zapisany w pliku „dane.txt” (z poprzedniego ćwiczenia)

Operacje na katalogach

Za pomocą funkcji mkdir() można tworzyć nowe katalogi

mkdir(‚nazwa_katalogu’)

Funkcja zwraca wartość true, jeżeli katalog został utworzony, lub wrtość false, jeżeli katalogu nie udało się utworzyć.

Katalog można usunąć:

rmdir(‚nazwa_katalogu’)

W wyniku wykonania funkcji zostanie usunięty katalog o podanej nazwie pod warunkiem, że jest pusty.
Podobnie jak w funkcji mkdir() zwraca true lub false.

Operacje na katalogach

Do otworzenia katalogu służy funkcja:

opendir(‚nazwa_katalogu’)

Funkcja zwraca deskryptor katalogu lub wartość *false*, jeżeli katalogu o podanej nazwie nie uda się otworzyć.

Do odczytu zawartości katalogu stosowana jest funkcja

readdir(deskryptor)

Każde wywołanie funkcji powoduje zwrócenie nazwy kolejnego elementu znajdującego się w katalogu.Po osiągnięciu końca katalogu funkcja zwróci wartość false.

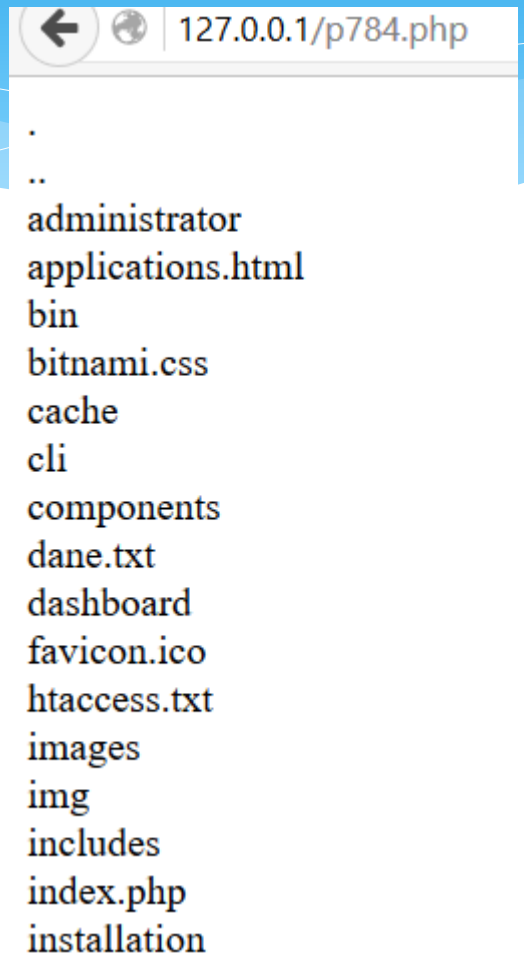
Do zamykania katalogu służy funkcja

closedir(deskryptor)

Funkcja nie zwraca żadnej wartości.

Przykład

```
<?php
$katalog="./";
if($deskr=opendir($katalog)){
while(($plik=readdir($deskr))!==false)
echo "$plik<br/>";
closedir($deskr);
}
else{
echo("Nie można otworzyć katalogu");
}
?>
```



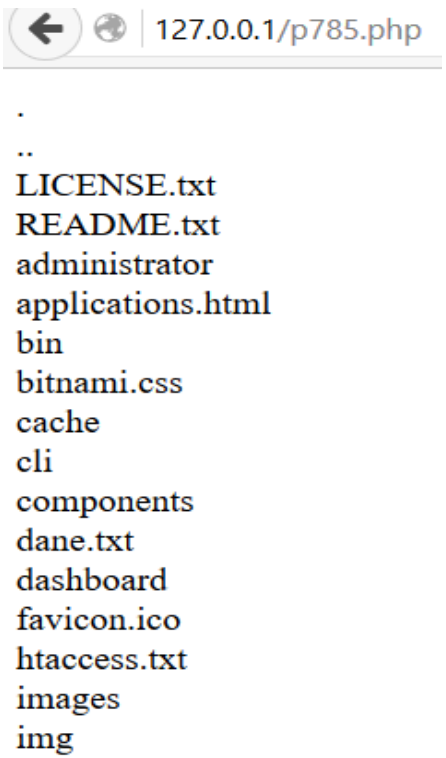
Przykład 2

Do odczytu zwartości katalogu może również zostać wykorzystana funkcja:

scandir(‚nazwa katalogu’)

Która pobiera zawartość całego katalogu i zapisuje ją w tablicy. Dodatkowo sortuje odczytane nazwy plików i folderów.

```
<?php
$katalog="./";
$tablica= scandir($katalog);
foreach($tablica as $plik){
        echo("$plik<br/>");
}
?>
```



Praktyczna zastosowanie

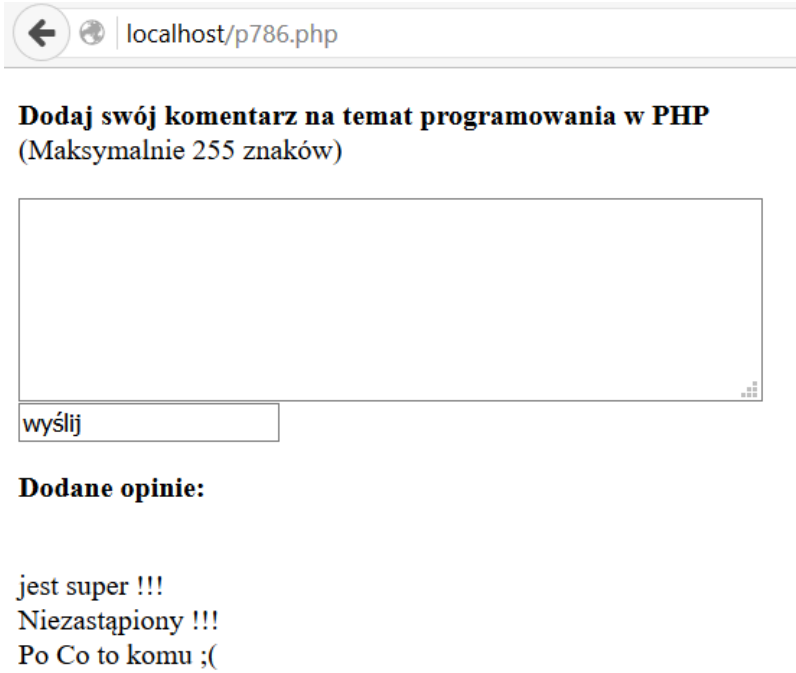
Przykładowy skrypt będzie zapisywał opinie użytkowników w pliku tekstowym opinie.txt

```
<?php
if(isSet($_POST['komentarz'])){
		$tekst=substr($_POST['komentarz'],0,255);
		$tekst=strip_tags($tekst)."\n";
		if(!$op=fopen('opinie.txt','a')){
			echo("Błąd. Nie można otworzyć pliku");   }
		else{
		if(fwrite($op, $tekst)===false){echo("dodanie komentarza nie powiodło się");}
		}}?>
<html><head>
		<title>Opinie użytkowników</title>
		<meta http-equiv="Conten-Type" content="text/html; charset=UTF-8">
		</head>
<body>
<div>
<form action="http://localhost/p786.php" method="post">
<p><b>Dodaj swój komentarz na temat programowania w PHP</b><br/>
(Maksymalnie 255 znaków)</p>
<textarea name="komentarz" rows="6" cols="50" wrap="virtual"></textarea><br/>
<input tape="submit" value="wyślij"></div></form>
		<p><b>Dodane opinie:</b></p><br/>
		<div>
```

cd..

```
<?php
$opinie='';
if(file_exists('./opinie.txt')){

        $opinie=file_get_contents('./opinie.txt');
        $opinie=nl2br($opinie);        }
if($opinie!=''){
                echo($opinie);        }
else{
        echo("Brak opinii");
}
?>
</div>
</body>
</html>
```



Wykonaj samodzielnie

Napisz skrypt, który będzie odczytywał zawartość wskazanego katalogu, a odczytane elementy podzieli na katalogi i pliki, wyświetli je w kolejności: wszystkie katalogi, następnie wszystkie pliki. Wykorzystaj funkcje *is_file()* oraz *is_dir()*.

Funkcje wyjścia

Do natychmiastowego zakończenia wykonania skryptu stosowane są instrukcje:

exit([argument])

die([argument])

Przykład

```
<?php
if(!$p=fopen(‚dane.txt’,’r’)) {
exit(„Nie można otworzyć pliku”);
else
{
// cd dalszy….
}
?>
```

Formularze HTML

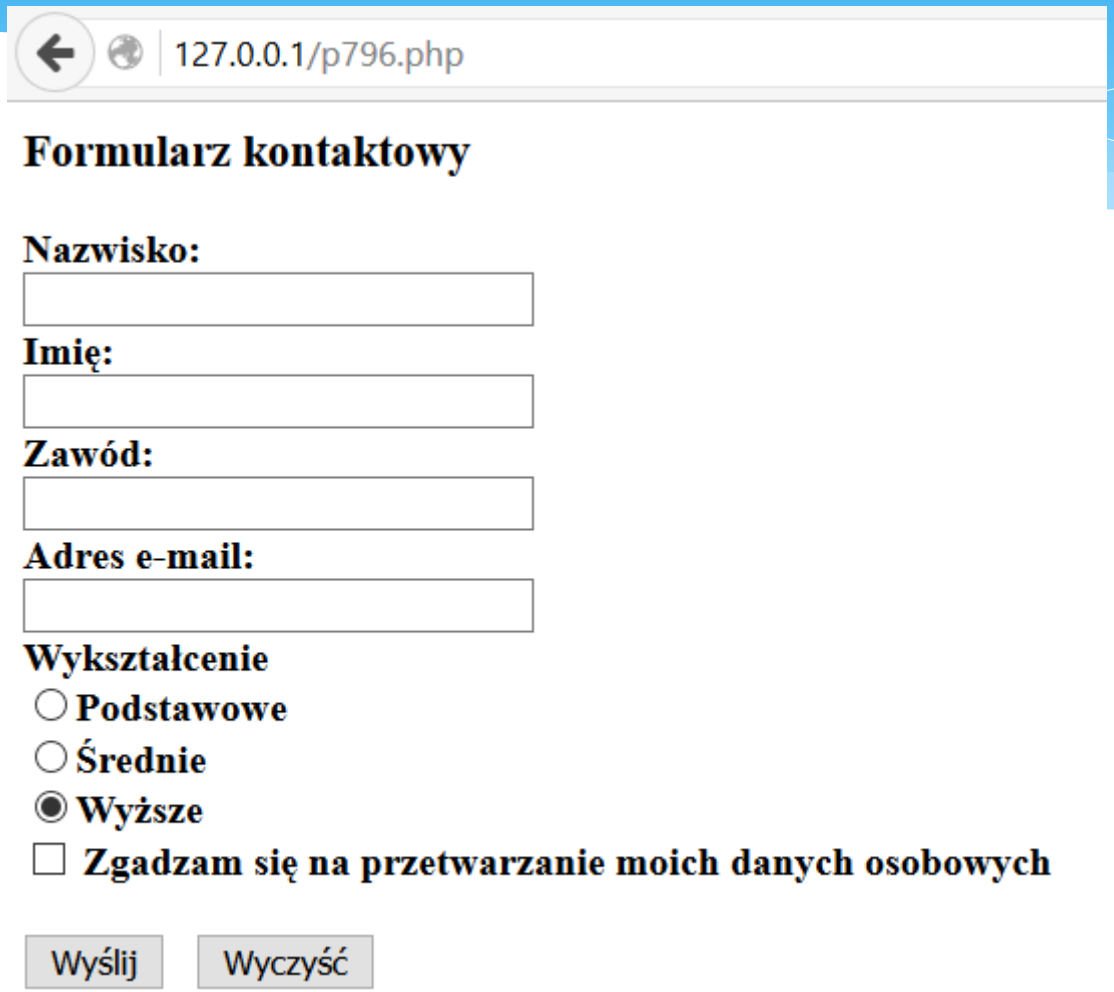
Przekazywanie danych z formularza kiedy danych jest niewiele wykorzystujemy metodę GET. W tej metodzie parametry przekazywane są za pomocą adresu URL.

http://www.nazwa_serwera.pl/strona.php?parametr1=wartość1&parametr2=wartość2

Metoda POST do przekazywania parametrów wykorzystuje nagłówek strony. Metoda ta umożliwia przekazywanie większej ilości parametrów, a parametry nie są widoczne w pasku przeglądarki.

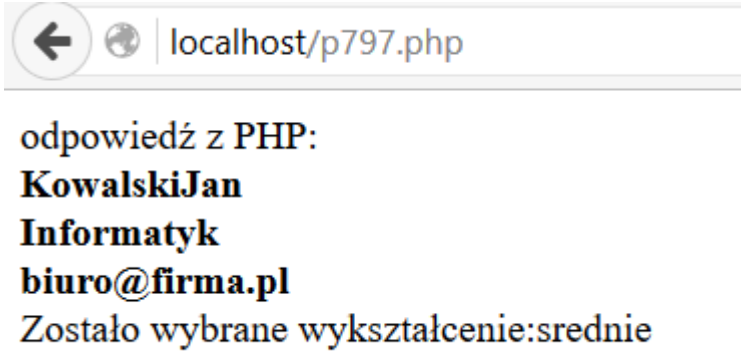
Przesyłane wartości parametrów są zapisywane w odpowiednich tablicach asocjacyjnych, odpowiednio $_GET i $_POST. Tablice te są superglobalne.

Wykonaj……


127.0.0.1/p796.php
Formularz kontaktowy
Nazwisko:
Imię:
Zawód:
Adres e-mail:
Wykształcenie
Podstawowe
Średnie
Wyższe
Zgadzam się na przetwarzanie moich danych osobowych
Wyślij
Wyczyść

Sprawdzamy czy działa

```
<?php
echo "odpowiedź z PHP:
<b><br/>".$_POST['nazwisko']."</b>";
echo "<b>".$_POST['imie']."</b><br/>";
echo "<b>".$_POST['zaw']."</b><br/>";
echo "<b>".$_POST['adr']."</b><br/>";
echo "Zostało wybrane
wykształcenie:".$_POST['wykszt'];
?>
```



Funkcja isSet

Pozwala sprawdzić czy pole zostało wypełnione

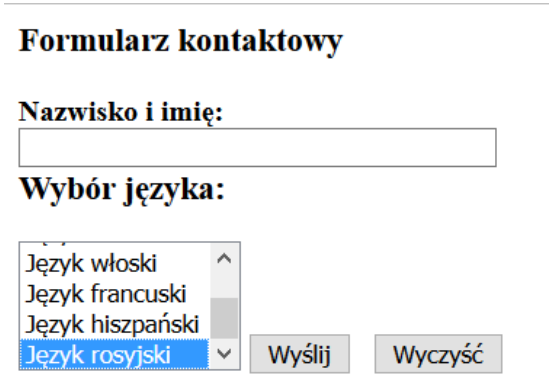
```
<?php
echo "odpowiedź z PHP:
.
.
.
if(!isSet($_POST['wykszt'])){
        echo "<br/>Proszę zaznaczyć pole wykształcenie";
}
else{
echo "Zostało wybrane wykształcenie:".$_POST['wykszt'];
}
?>
```

Funkcja empty

Sprawdza, czy istniejąca zmienna użyta jako argument jest pusta. Jeżeli jest pusta zwraca wartość *true*.

Przetestujemy to na przykładzie wyboru z listy kilku języków obcych.

```
<?php
if(!empty($_POST['języki'])){
        echo "<p><b>".$_POST['nazwisko']."</b> zna:</p>";
        echo "<ul>";
        foreach($_POST['języki'] as $wartosc){
                echo "<li>".$wartosc."</li>";
        }
        echo "</ul>";}
        else{echo „<p><b>".$_POST['nazwisko']."</b> nie zna żadnego języka";}
?>
```



Formularz językowy

```
<html><head>
<title>Formularz rejestracyjny</title>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
</head>
<body>
<form action="http://localhost/p799a.php" method="post">
<b><font size="4"><b>Formularz kontaktowy</b></font><br/><br/>
Nazwisko i imię:<br/>
<input type="text" name="nazwisko" value=" " size="40"><br/>
<b><font size="4"><b>Wybór języka:</font></b><br/><br/>
<select name ="języki[]" multiple>
<option value="Język angielski">Język angielski</option>
<option value="Język niemiecki">Język niemiecki</option>
<option value="Język włoskii">Język włoski</option>
<option value="Język francuski">Język francuski</option>
<option value="Język hiszpańskii">Język hiszpański</option>
<option value="Język rosyjski">Język rosyjski</option>
</select>
<input type="submit" value="Wyślij" name="wyslij">  
<input type="reset" value="Wyczyść" name="resetuj">
</form></body></html>
```

Ćwiczenie

Opracuj formularz przeznaczony do rejestrowania kandydatów na stanowisko informatyka w nowopowstałej firmie. Przygotuj formularz i skrypt PHP, który przetworzy dane przesłane z formularza. W skrypcie powinno być sprawdzane wypełnienie wszystkich pól.

Przesyłanie plików na serwer

Formularze służące do przesyłania plików w definicji znacznika <form> muszą zawierać atybut:

enctype – określa sposób kodowania danych; musi zostać podana wartość: ***enctype=„multipart/form-data***

action – określa adres skryptu PHP

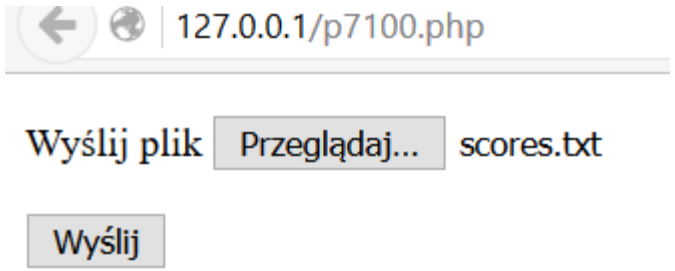
method – określa metodę wysyłania danych; musi to być metoda POST

Do wysyłania pliku służy pole *input* typu *file*. Można dodać ukryte pole o nazwie *max_file_size* w celu określenia maksymalnego rozmiaru pliku, który można będzie przesłać na serwer w postaci:

<input type=„hidden” name=„max_file_size” value =„wielkość_pliku”>

Skrypt formularza

```
<html>
<head>
<title>Formularz - przesyłanie plików</title>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
</head>
<body>
<form enctype="multipart/form-data" action="odbierz_plik.php"
method="post">
<input type="hidden" name="max_file_size" value="40960">
<p>Wyślij plik
<input type="file" name="plik1" size="40"></p>
<p><input type="submit" value="Wyślij" name="wyslij"></p>
</form>
</body>
</html>
```



Odbieramy plik

Informacje o plikach, które zostały wysłane, są przechowywane w superglobalnej tablicy asocjacyjnej ***$_FILES***. Indeksami tej tablicy są nazwy pól ***input*** typu ***file*** pobrane z formularza.

$_FILES[‚nazwa_pliku’][‚name’]- nazwa przesłanego pliku

$_FILES[‚nazwa_pliku’][‚type’]-typ MIME pliku

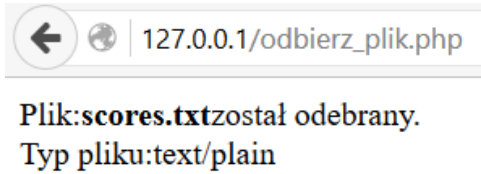
$_FILES[‚nazwa_pliku’][‚size’]-wielkość pliku w bajtach

$_FILES[‚nazwa_pliku’][‚tmp_name’]-ścieżka tymczasowa, pod którą plik został zapisany na serwerze

$_FILES[‚nazwa_pliku’][‚error’]-kod błędu

Move_uploaded_file() i is_uploaded_file() czy plik tymczasowy istnieje

```
<?php
$katalog_plik="./";
$max_rozm=$_POST['max_file_size'];
$zm=$_FILES['plik1']['tmp_name'];
if (is_uploaded_file($zm)){
if ($_FILES['plik1']['size']>$max_rozm){
echo "Błąd. Plik ma za duży rozmiar";
}
else{
echo "Plik:<b>".$_FILES['plik1']['name']."</b>został odebrany.</br>";
if(isSet($_FILES['plik1']['type']))
echo "Typ pliku:".$_FILES['plik1']['type']."<br/>";
move_uploaded_file($_FILES['plik1']['tmp_name'],$katalog_plik.$_FILES['plik1']['na
me']);
}}
else
echo "Błąd przesyłania pliku";
?>
```



Ćwiczenie

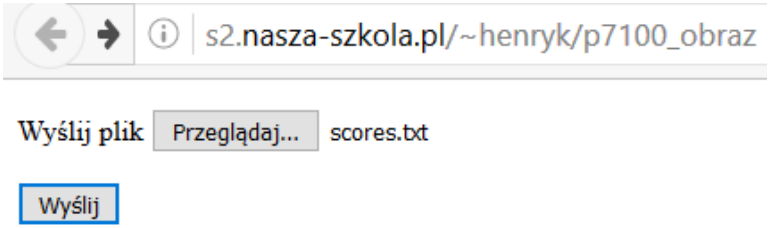
Do opracowanego w poprzednim ćwiczeniu formularza dodaj pole umożliwiające przesłanie pliku ze zdjęciem. Przyjmij, że przesłane pliki mogą być w formacie JPG, JPEG lub PNG. Ich rozmiar nie może przekroczyć 900kB. Zmodyfikuj skrypt PHP tak, aby możliwe było odbieranie pliku przesłanego przez formularz.

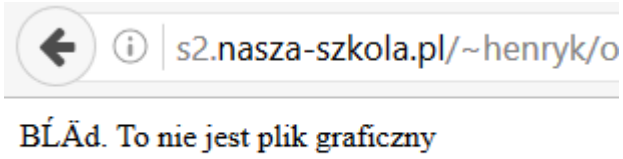
Mała wskazówka

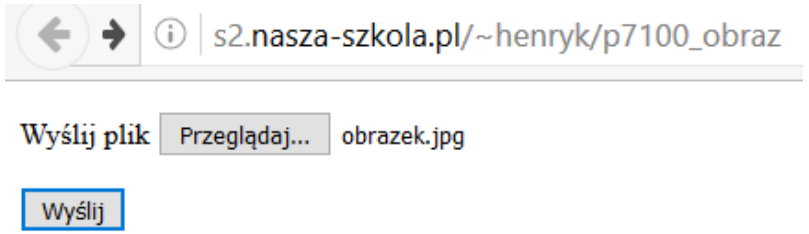
$_FILES[‚nazwa_pliku’][‚type’]-typ MIME pliku

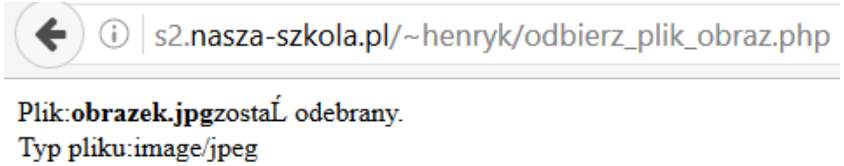
Dla plików tekstowych przyjmuje wartość text/plain, w przypadku plików graficznych może być:

image/jpeg lub image/png









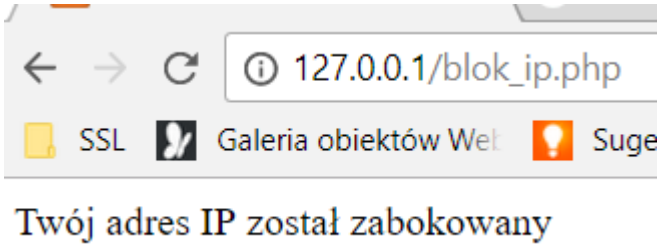
Przykładowe rozwiązanie

```
<?php
$typ=$_FILES['plik1']['type'];
if ($typ != 'image/jpeg')
{               echo "Błąd. To nie jest plik graficzny";}
else{
$katalog_plik="./";
$max_rozm=$_POST['max_file_size'];
$zm1=$_FILES['plik1']['size'];
$zm=$_FILES['plik1']['tmp_name'];
if (is_uploaded_file($zm)){
if ($_FILES['plik1']['size']>$max_rozm){
echo "Błąd. Plik ma za duży rozmiar";}
else{
echo "Plik:<b>".$_FILES['plik1']['name']."</b>został odebrany.</br>";
if(isSet($_FILES['plik1']['type']))
echo "Typ pliku:".$_FILES['plik1']['type']."<br/>";
move_uploaded_file($_FILES['plik1']['tmp_name'],$katalog_plik.$_FILES['plik1']['name']);
}}
else
echo "Błąd przesyłania pliku";}
?>
```

Praktyczne skrypty
blokowanie adresu IP

Adres IP z którego nastąpiło połączenie z witryną jest zapisany w kluczu REMOTE_ADDR. Jego znajomość pozwala np. na blokowanie dostępu do naszej strony wybranym adresom IP, lub odwrotnie wpuszczenie tyko ze znanych nam adresów.

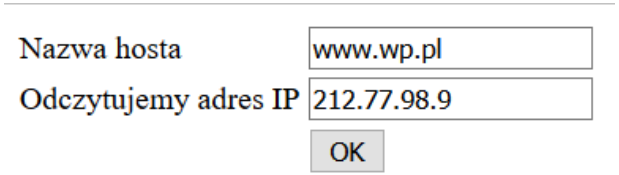
```
<?php
$ip_tab=file("ip.txt");
foreach($ip_tab as $b){
if (trim($b)==$_SERVER['REMOTE_ADDR']){
echo "Twój adres IP został zablokowany";
exit;
}}
?>
```



Lista blokowanych adresów IP zapisana jest w pliku ip.txt

Praktyczne skrypty
uzyskanie adresu IP

```
<?php
$ip="";
$host=@$_POST["host"];
if($host!="")
$ip=gethostbyname($host);
$html_code=<<<CODE
<html>
<head>
<title>Pozyskanie adresu IP</title></head>
<body>
<form name="formularz"
action="http://127.0.0.1:80/ip.php" method="POST">
<table><tr><td>Nazwa hosta</td>
<td><input type="text" name="host"
value="$host"></td>
</tr><tr><td>Odczytujemy adres IP</td>
<td><input type="text" name-"ip" value=$ip></td>
</tr><tr><td></td>
<td align="left">
<input type="submit" name="wyślij" value="OK">
</td></tr>
</table></body></html>
CODE;
print("$html_code");
?>
```



Połączenie FTP

W języku PHP można zrealizować obsługę protokołu FTP.

Przed wysłaniem lub pobraniem pliku należy nawiązać połączenie z serwerem FTP za pomocą funkcji :

ftp_connect(nazwa_hosta[,port[,timeout]])

Argumenty port i timeout są opcjonalne. Jeżeli nie zostaną zdefiniowane to port zostanie domyślny 21, timeout:90. Funkcja zwraca wartość *false* jeżeli nie uda się nawiązać połączenia

Połączenie FTP

Po nawiązaniu połączenia kolejnym działaniem jest zalogowanie się.

ftp_login(ftp_id, nazwa_użytkownika, hasło)

ftp_id – to identyfikator połączenia zwrócony przez funkcję *ftp_connect.*

Logowanie powinno zostać wykonane również wtedy, gdy połączenie dotyczy serwerów oferujących dostęp anonimowy.

Podajemy wówczas *nazwa_użytkownika:anonymous*, a *hasło* to adres e-mail.

Połączenie FTP

Pobieranie pliku można zrealizować za pomocą funkcji:

ftp_fget(ftp_id,plik_lokalny, plik_zdalny, tryb[, pozycja])

ftp_id- identyfikator z funkcji *ftp_connect()*

plik_lokalny to nazwa pliku do którego pobierane będą dane.

Plik_zdalny – nazwa pliku do pobrania (ze scieżką dostępu)

Tryb – FTP_ASCI (transmisja tekstowa), FTP_BINARY (transmisja binarna)

Pozycja – określa po wznowieniu przerwanej transmisji pozycję pliku(argument opcjonalny)

Po zakończeniu transmisji połączenie powinno zostać zamknięte

ftp_close(ftp_id)

Przykład FTP

```
<?php
$serwer="127.0.0.1";
$user=,,username";
$pass=,,*********";
$plik_lok="plik.txt";
$plik_zdal="plik1.txt";
$dir="\dane";
//if(!$id=ftp_ssl_connect($serwer)){
if(!$id=ftp_connect($serwer,21)){
exit("Błąd połączenia z serwerem");}
        if(!ftp_login($id, $user, $pass)){
                exit("Błąd logowania do serwera");}
        else{echo "Połączenie nawiązane.<br/>";   }
        if (!ftp_get($id, $plik_lok, $plik_zdal, FTP_BINARY))
                {    exit("Błąd podczas pobierania pliku");    }
        else{
                echo "Plik $plik_lok został pobrany";  }
ftp_close($id);

?>
```

UWAGA !!!
W XAMP musisz dodać użytkownika w FileZilla

Praktyczne skrypty

Rekordy MX

Rekord MX

Rekord wskazuje komputery, które mogą przejąć (czasowo lub na stałe) pocztę kierowaną na określony adres.

```
<html><head>
<title>Rekordy MX</title></head>
<body><form name="formularz"
action="http://127.0.0.1:80/mx.php"
method="POST">
<table>
<tr><td>Nazwa hosta</td>
<td><input type="text" name="host"
value=""></td>
<td align="left"><input type="submit"
name="wyślij" value="Sprawdz rekordy
MX">
</td></tr></table></body></html>
```

Rekordy MX dla hosta wp.pl
mx5.wp.pl
mx.wp.pl

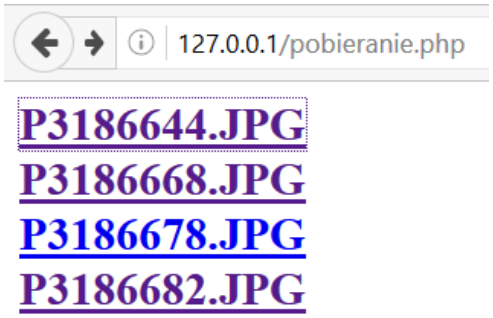
```
<?PHP
@$mxtab;
$host=$_POST["host"];
if(@getmxrr($host,$mxtab)){
        print("Rekordy MX dla hosta $host<br>");
        foreach($mxtab as $value){
                print("$value<br>");  }  }
else{
        print("nie można odczytać rekordów MX");  }
?>
```

Praktyczne skrypty

Lista plików do pobrania

Do realizacji potrzebne będą instrukcje *opendir()*, pętla while i funkcja *readdir()*. Po odczytaniu listy plików wykorzystamy znacznik <A>

```
<html>
<head><title>Pobieranie plikow</title></head>
<body><h2>
        <?PHP
        $dir="photo/";
        function printDir($dir)
        {$fd=opendir($dir);
                if(!$fd) return false;
                while(($file=readdir($fd))!==false){
                        if($file!="."&& $file !=".."){
                                echo("<a href=$dir$file>$file</a><br>"); } }
                closedir($fd);          }
        printDir($dir);
        ?>
        </h2></body></html>
```



Pliki cookies i sesje

Pliki cookies w PHP

Pliki cookies to niewielkie pliki tekstowe wysyłane przez serwer lub skrypt do przeglądarki i umieszczane przez nią na dysku użytkownika. Pliki te są częścią specyfikacji protokołu HTTP i są wysyłane do przeglądarki w postaci nagłówka o nazwie *Set-Cookie.* Plik zawiera między innymi nazwę serwera, datę wygaśnięcia oraz informację na temat domeny i ścieżki. Rozmiar nie może przekraczać 4 kB.

Zasada działania:

* Po nawiązaniu połączenia serwer wysyła do przeglądarki nagłówek *Set-Cookie*, który zawiera plik cookie
* Przeglądarka zapisuje plik na dysku
* Przy kolejnym połączeniu z serwerem przeglądarka wysyła do niego przechowywany na dysku plik cookie

Pliki cookies i sesje

Tworzenie pliku cookie

W skryptach PHP pliki cookies tworzone są za pomocą funkcji setcookie(), która ma postać:

setcookie(nazwa[,wartość[,czas_trwania[,ścieżka_dostępu[,domena[,bezpieczeństwo[,tylko_http]]]]]])

Jedynym argumentem, który musi wystąpić w funkcji, jest nazwa, pozostałe są opcjonalne.

Funkcja setcookie musi być wywołana w kodzie przed znacznikiem <HTML>

```
<?php
setcookie("pismo", "Na skróty", time()+3600, "/", "localhost",false);
if(isSet($_COOKIE["pismo"]))
echo "<p>Jesteś naszym stałym klientem</p>";
else
echo "<p>Witamy po raz pierwszy na naszej stronie.</p>";
?>
```

Pliki cookies i sesje

Usuwanie pliku cookie

Jeżeli dla pliku cookie został ustalony czas ważności, to zostanie on automatycznie usunięty przez przeglądarkę po określonym czasie. Gdy czas ważności nie był określony, to plik zostanie usunięty przy zamykaniu przeglądarki. Aby usunąć samodzielnie plik cookie, można ustawić dla niego czas ważności, który już upłynął, albo ustawić jako wartość ciąg pusty lub wartość false.

```
setcookie("pismo", "Na skróty", time()-100, "/", "localhost",0);
```

Pliki cookies i sesje

Data ostatnich odwiedzin

Wykorzystując plik cookie, określamy datę ostatnich odwiedzin strony przez użytkownika;

```
<?php
$mies=2592000+time();
setcookie("wizyta",date("F jS -q:ia"),$mies);
?>
```

Zostanie utworzony plik cookie wizyta. Termin wygaśnięcia to 30 dni od chwili jego utworzenia (60s*60min*24godz*30dni=2 592 000)

Plik tworzący skrypt odczytujący zawartość pliku cookie;

```
<?php
if(isSet($_COOKIE['wizyta']))
{
$ostatnia=$_COOKIE['wizyta'];
echo "Witamy ponownie<br>Ostatni raz odwiedziłeś
nas:".$ostatnia;}
else{
echo"Witamy na naszej stronie";}
?>
```

Sesje

Mechanizm sesji dostępny w języku PHP pozwala na identyfikację użytkownika i śledzenie jego poczynań na stronie internetowej. W momencie pierwszego wejścia na stronę tworzony jest dla użytkownika specjalny identyfikator. Jest on przechowywany na komputerze użytkownika (w pliku cookie) i na serwerze. Wszystkie zmienne związane z użytkownikiem są przechowywane na serwerze w tablicy superglobalnej $_SESSION.

Do rozpoczęcia sesji najczęściej jest wykorzystywana funkcja

session_start();

W każdym skrypcie, który będzie korzystał z sesji, należy wywołać tę funkcję. Jeżeli wszystkie skrypty będą korzystały z sesji można zmienić w pliku *php.ini* wartość opcji *session.auto_start na 1.*

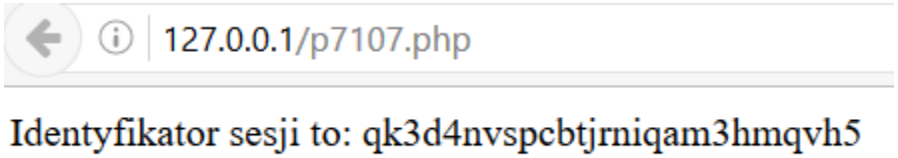
Sesje

Do zakończenia sesji należy wykorzystać funkcję

session_destroy();

Identyfikator sesji można odczytać za pomocą funkcji session_id().

Testujemy




```
<?php
session_start();
echo "<p>Identyfikator sesji to: ".session_id()."</p>";
?>
```

Podany kod wyświetli informację na temat wartości identyfikatora. Przy pierwszych odwiedzinach strony PHP wygeneruje dla użytkownika nowy identyfikator. Przy kolejnych przydzieli mu ten sam.

Zmienne sesji

Po rozpoczęciu sesji w tablicy $_SESSION mogą zostać zapisane dowolne zmienne związane z sesją. Zapisujemy zmienną używając konstrukcji:

$_SESSION[,nazwa_zmiennej']=wartość;

Do odczytania wartości trzeba użyć indeksu odnoszącego się do nazwy zmiennej w tablicy

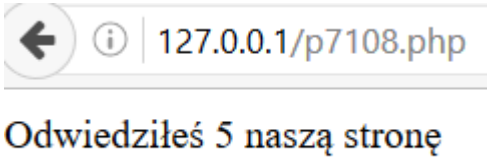
$zmienna=$_SESSION[,nazwa_zmiennej,];

Do sprawdzenia, czy określona zmienna istnieje, można zastosować funkcję isSet(), która zwróci true jeżeli zmienna została zarejestrowana.

Zmienne sesji

Zmienne utworzone w sesji są przechowywane w pliku tymczasowym. Dodanie do tablicy $_SESSION powoduje dopisanie jej nazwy i wartości do tego pliku. Zapisane dane mogą zostać odczytane przez inne skrypty w obrębie tej samej sesji.

```
<?php
session_start();
if(!isSet($_SESSION['ile'])){
        $_SESSION['ile']=0;
}
$_SESSION['ile']++;
echo "<p>Odwiedziłeś ".$_SESSION['ile']." naszą stronę</p>";
?>
```



Uwierzytelnianie użytkownika

Mechanizm sesji wykorzystamy do przeprowadzenia autoryzacji użytkownika.

Przygotujemy 3 pliki. Plik *strona.php* będzie wyświetlał stronę główną witryny, *loguj.php* zawiera moduł autoryzacji użytkownika poprzez formularz. Plik *wyloguj.php* zawiera procedury wylogowania.

Skrypt *loguj.php*

```
<?php
session_start();
if(isSet($_SESSION['log'])){
header('location:strona.php');
exit();
}
else if (isSet($_POST['nazwa'])&& isSet($_POST['haslo'])){
		if($_POST['nazwa']=='janek'&& $_POST['haslo']=='1234'){
			$_SESSION['log']='nazwa';
			exit();
		}
		else{
			echo "nieprawidłowe dane logowania";
		}
}
?>
<!DOCTYPE HTML>
<html>
		<head>
		<title>Autoryzacja danych</title>
		<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
		</head>
		<body>
			<div>
			<form action="http://127.0.0.1/loguj.php" method="post">
			<b>Logowanie</b><br><br>
			Nazwa użytkownika:<br>
			<input type=text name="nazwa" value=""size="25"><br>
			Hasło:<br>
			<input type=password name='haslo' value="" size="25">
			<input type="submit" value="Zaloguj się">
			</form>
			</div>
		</body>
</html>
```

Skrypt *strona.php*

```
<?php
session_start();
if(isSet($_SESSION['log'])){
header('location:loguj.php');
exit;
}
?>
<!DOCTYPE HTML>
<html>
        <head>
        <title>Strona główna</title>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        </head>
        <body>
                <p>Jesteś na stronie głównej</p>
                <p>Przed opuszczeniem strony wyloguj się!</p>
                <a href = "wyloguj.php">Wyloguj</a>
        </body>
</html>
```

Skrypt *wyloguj.php*

```
<?php
session_start();
if(isSet($_SESSION['log'])){
unset($_SESSION['log']);
}
else{
header('location:loguj.php');
exit;
}
$s=session_destroy();
?>
<!DOCTYPE HTML>
<html><head>
<title>Koniec sesji</title>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
</head>
                    <body>
                        <p>Wylogowałeś się ze strony</p>
                        <a href = "loguj.php">Logowanie</a>
</body></html>
```

Zadania

* Zmodyfikuj skrypt loguj.php tak, aby weryfikacja danych dotyczyła wielu użytkowników. Utwórz tabelę ‚użytkownicy', która będzie zawierała loginy oraz hasła użytkowników korzystających z danej strony internetowej.
* Przy założeniu, że Twoja witryna internetowa składa się z kilku stron, napisz kod, który pozwoli na dostęp do stron witryny tylko ze strony głównej.

Połączenie z bazą danych

http://localhost/phpmyadmin w xampp lub

http://s2.nasza-szkola.pl/myadmin/

Bazy danych w PHP

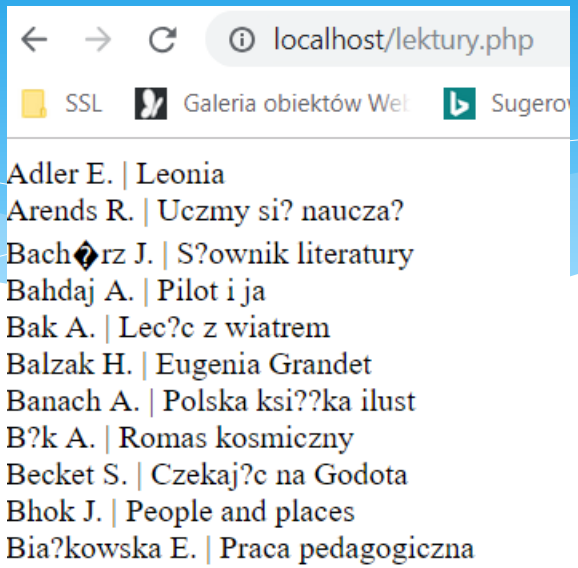
- Kod skryptu rozpoczynamy od połączenia z bazą danych za pomocą funkcji **mysql_connect.** Następnie ustalamy bazę danych, z którą będziemy pracowali. Zadanie to realizuje funkcja **mysql_select_db**, której parametrem jest nazwa bazy danych. Następnie zadajemy zapytanie SQL, którego rezultatem będzie zawartość tabeli : **"SELECT * FROM tabela".** Wyniki zwrócone przez zapytanie przetwarzamy w pętli while. Funkcja **mysql_fetch_array** zwraca jako wynik kolejny rekord będący rezultatem wydanego zapytania SQL. Parametrem funkcji mysql_fetch_array jest wynik zwrócony przez funkcję mysql_query. Zmienna $row będąca wynikiem zwróconym przez funkcje mysql_fetch_array jest tablicą asocjacyjną, która zawiera kolejne pola danego rekordu.

```php
<?php
$link = mysqli_connect("localhost", "root", "")
   or die("brak połączenia z serwerem");

mysqli_select_db($link,"henryks")
   or die("błąd bazy danych");

$query  = "SELECT * FROM lektury";
$result = mysqli_query($link,$query)
   or die("brak tabeli");

while ($row = mysqli_fetch_array($result)) {
   echo "<TR><TD>" . $row["autor"] . "</TD><TD>" ." | ". "</TD><TD>" .
$row["tytul"] .
       "</TD></TR><BR>";
}
mysqli_free_result($result);
mysqli_close($link);
?>
```



Klauzula While

W klauzuli **WHERE** formułuje się warunek, który odpowiada warunkowi wyboru (selekcji) w algebrze relacyjnej i który określa ograniczenia, jakie mają spełniać rekordy, aby zostać wybrane w danym zapytaniu.

Jeżeli rekord spełnia te ograniczenia to zostaje dołączony do tabeli wynikowej.

Postać zapytania:

SELECT *
FROM *nazwa-tabeli*
WHERE *warunek*;

Klauzula WHERE pozwala na wybranie z tabeli tych wierszy, które spełniają określone warunki:

SELECT *
FROM NAZWISKA
WHERE STANOWISKO = 'INFORMATYK';

Dla podanego przykładu z tabeli zostaną wybrane tylko te rekordy, w których w polu STANOWISKO jest wpisane 'INFORMATYK'

Formułowanie warunku

Po słowie kluczowym **WHERE** występuje wyrażenie warunkowe.

Do zapisu porównywania wartości w języku SQL służy sześć operatorów:

równy **=**

nierówny **<>**

mniejszy **<**

większy **>**

mniejszy lub równy **<=**

większy lub równy **>=**

W wyrażeniu mogą występować stałe oraz nazwy kolumn tabel wymienionych w klauzuli **FROM**

Formułowanie warunku

Dla wartości numerycznych można budować wyrażenia arytmetyczne korzystając z operatorów + - * / i nawiasów ()

Stałe tekstowe w SQL są ujmowane w pojedyncze cudzysłowy.

'Przykładowe stałe'

W wyniku porównania powstaje wartość logiczna **TRUE (prawda) lub FALSE (fałsz)**

Wartości logiczne można łączyć w wyrażenia logiczne za pomocą operatorów logicznych **AND, OR i NOT**

Priorytet operatorów wykorzystywanych w budowie wyrażeń : operatory porównania, **NOT, AND, OR**

Przykład

```
$query  = "SELECT * FROM lektury WHERE
autor='Brzechwa J.'";
```

SSL Galeria obiektów We

Brzechwa J. | Akademia Pana Kleksa
Brzechwa J. | Pch?a Szachrajka
Brzechwa J. | Brzechwa dzieciom
Brzechwa J. | Brzechwa dzieciom
Brzechwa J. | Brzechwa dzieciom
Brzechwa J. | Wiersze

Lub

```
$query  = "SELECT * FROM lektury WHERE
autor='Brzechwa J.' AND tytul='Brzechwa dzieciom'";
```

SSL Galeria obiektów We

Brzechwa J. | Brzechwa dzieciom
Brzechwa J. | Brzechwa dzieciom
Brzechwa J. | Brzechwa dzieciom

WHERE z operatorem LIKE

Działa na kolumnach zawierających wartości łańcuchowe.

Operator LIKE sprawdza czy wartość tekstowa odpowiada podanemu wzorcowi, umożliwia więc wykonywanie częściowych porównań, takich jak „zaczynający się od tekstu”, „kończący się na tekście”, lub „zawierający tekst”

Tworząc wzorce stosuje się znaki wieloznaczne:

% - **zastępuje** sekwencję dowolnych znaków o długości n (gdzie n może być zerem)

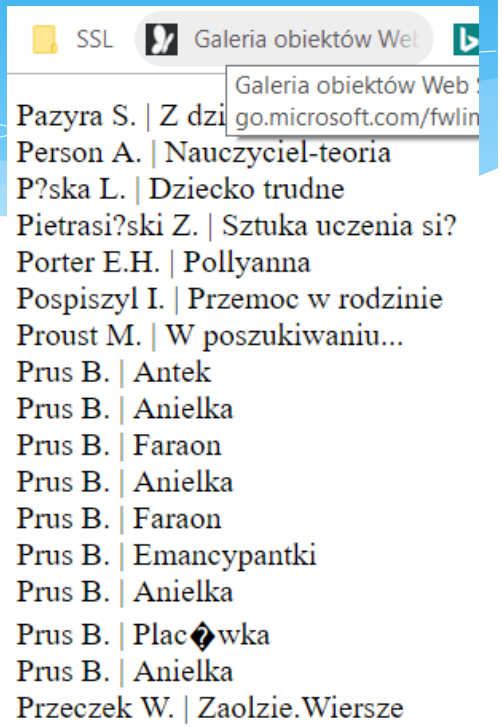
_ - **odpowiada jednemu** znakowi w przeszukiwanym tekście

WHERE z operatorem LIKE

Ogólna postać polecenia z operatorem **LIKE:**

"SELECT * FROM lektury WHERE autor LIKE 'P%',,

Zwróci wszystkich autorów których nazwiska zaczynają się na literę P.



Przykład (autorzy na literę P)

```
<?php
$link = mysqli_connect("localhost", "root", "")
   or die("brak połączenia z serwerem");

mysqli_select_db($link,"henryks")
   or die("błąd bazy danych");

$query  = "SELECT * FROM lektury WHERE autor LIKE 'P%'";

$result = mysqli_query($link,$query)
   or die("brak tabeli");

while ($row = mysqli_fetch_array($result)) {
   echo "<TR><TD>" . $row["autor"] . "</TD><TD>" ." | ". "</TD><TD>" .
$row["tytul"] .
       "</TD></TR><BR>";
}
mysqli_free_result($result);
mysqli_close($link);?>
```

Przykłady teoretycznie

* Załóżmy, że chcesz wyszukać pracowników w tabeli *pracownicy*, których imię rozpoczyna się znakiem 'a', możemy to zrobić w następujący sposób:

```
SELECT numer_pracownika, nazwisko, imie
FROM pracownicy
WHERE imie LIKE 'A%'
```

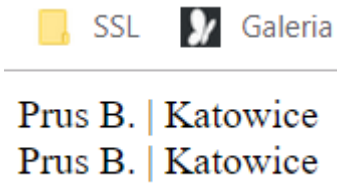
* Aby wyszukać wszystkich pracowników, których nazwiska kończą się na ‚on' można wykonać zapytanie w następujący sposób:

```
SELECT numer_pracownika, nazwisko, imie
FROM pracownicy
WHERE imie LIKE '%on'
```

Przykłady

Poszukamy autora, którego nazwisko zaczyna się na literę „P”, i wydał książki w mieście które kończy się na „ce”.

= "SELECT * FROM lektury WHERE autor LIKE 'P%' AND miejsce LIKE '%ce'";



Przykłady teoretycznie

Jeśli chcesz odszukać dany ciąg znaków, który znajduję się w środku słowa, możesz umieścić znak procentu na początku i na końcu, aby znaleźć poszukiwany ciąg znaków.

```
SELECT numer_pracownika, nazwisko, imie
FROM pracownicy
WHERE imie LIKE '%on%'
```

* Aby wyszukać wszystkich pracowników, których nazwiska są jak Byk, Bok. Możesz użyć znaku dolnego podkreślenia (_):

```
SELECT numer_pracownika, nazwisko, imie
FROM pracownicy
WHERE nazwisko LIKE 'B_k'
```

Przykłady

Poszukamy autora, którego nazwisko zaczyna się na literę „P”, i wydał książki w mieście które kończy się na „wa”, a druga litera w tytule to „n”

= "SELECT * FROM lektury WHERE autor LIKE 'P%' AND miejsce LIKE '%wa' AND tytul LIKE '_n%'";

—

Prus B. | Warszawa | Antek
Prus B. | Warszawa | Anielka
Prus B. | Warszawa | Anielka

Przykłady

* MySQL wraz z operatorem LIKE umożliwia stosowanie przeczenia NOT, aby zaleźć wszystkie ciągi znaków, które nie są dopasowane do określonego wzorca. Załóżmy, że chcesz wyszukać wszystkich pracowników, których nazwisko nie zaczyna się od znaku 'B', możesz użyć następującego zapytania:

SELECT numer_pracownika, nazwisko, imie
FROM pracownicy
WHERE nazwisko NOT LIKE 'B%,

Operator LIKE nie rozróżnia wielkości liter, tak więc 'b%' i 'B%' są takie same.

Zadania (zapytania wybierające)

Pokaż książki autora którego nazwisko zaczyna się na literę **B**

Odpowiedź: 29 pozycji.

Pokaż książki droższe niż 5 zł, ale tańsze niż 10 zł.

Odpowiedź: 79 pozycji.

Pokaż książki **WYDANIE** 1 **lub** 3.

Odpowiedź: 138 pozycji.

Pokaż książki których **TYTUŁ** składa się z 5 liter.

Odpowiedź: np. Diuna

Pokaż książki których **TYTUŁ** kończy się pojedynczą literą.

Odpowiedź: np. Historia Polski XX w

Zapytania wstawiające dane

Do zapytań modyfikujących zawartość tabeli funkcja mysql_qwery() zwraca wartość true lub false. Za pomocą funkcji mysql_afected_rows() można odczytać, ile wierszy zostało zmodyfikowanych. Funkcja ma postać

mysqli_afected_rows([id])

id jest opcjonalny i zawiera identyfikator połączenia z serwerem.

Zapytanie dodające nowe dane do tabeli to polecenie **INSERT INTO.**

Zapytania wstawiające dane - przykład

```
<?php
$link = mysqli_connect("localhost", "root", "")
   or die("brak połączenia z serwerem");

mysqli_select_db($link,"henryks")
   or die("błąd bazy danych");

$dodaj="INSERT INTO lektury VALUES('','Lis M','101 praktycznych
skryptów','1','Gliwice','2018','56')";
if(!$zapytanie=mysqli_query($link,$dodaj)){
                        mysqli_close($link);
                        exit("Błąd w zapytaniu");}
mysqli_close($link);
?>
```

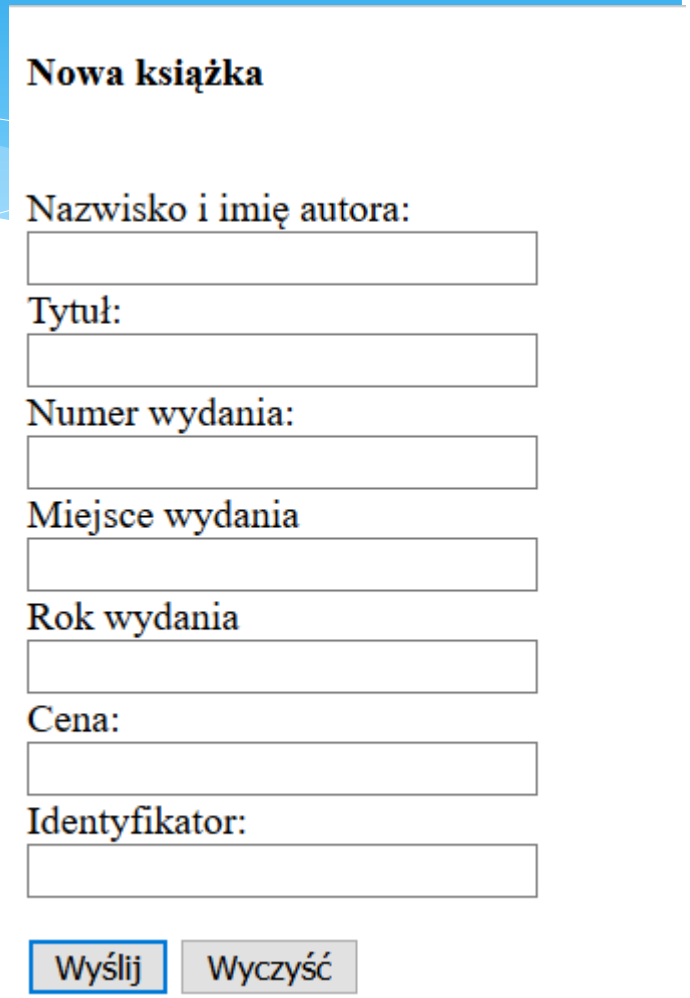
Zastosuj mysql_afected_rows() w celu sprawdzenia czy rekord został dodany

Dodawanie poprzez formularz

Myślę, że opracowanie formularza do wprowadzania nowych pozycji książkowych nie będzie dla Ciebie problemem.

Napisanie skryptu na podstawie wcześniejszego ćwiczenia także nie powinno być trudne

POWODZENIA.



Przykładowy formularz

```
<!DOCTYPE HTML>
<html lang="pl">
        <head><title>Dodawanie książek</title>
<META HTTP-EQUIV="content-type" CONTENT="text/html; charset=UTF-8"></head>
<body>
    <form action="http://localhost/dodaj_ksiazke.php" method="post">
    <p><b>Nowa książka</b></p><br/>
Nazwisko i imię autora:<br/>  <input tape="text" name="nazwisko" value=""
size="30"><br/>
Tytuł:<br/>  <input tape="text" name="tytul" value="" size="30"><br/>
Numer wydania:<br/>  <input tape="text" name="numer" value="" size="30"><br/>
Miejsce wydania<br/>  <input tape="text" name="miejsce" value="" size="30"><br/>
Rok wydania<br/>  <input tape="text" name="rok" value="" size="30"><br/>
 Cena:<br/>  <input tape="text" name="cena" value="" size="30"><br/>
 Identyfikator:<br/>  <input tape="text" name="id" value="" size="30"><br/>
        <p><input type="submit" value=„Dodaj" name="wyslij">
        <input type="reset" value="Wyczyść" name="zeruj"></p>
        </form>
        </body>
</html>
```

Przykładowy skrypt php

dodaj_ksiazke.php

```
<?php
$link = mysqli_connect("localhost", "root", "")
   or die("brak połączenia z serwerem");

mysqli_select_db($link,"henryks")
   or die("błąd bazy danych");

$dodaj="INSERT INTO lektury
VALUES('$_POST[nazwisko]','$_POST[tytul]','$_POST[numer]','$_POST[miejsce]','$_POST
[rok]','$_POST[cena]','$_POST[id]')";
$zapytanie=mysqli_query($link,$dodaj);
if($zapytanie==true){echo "Nowa książka została dodana";}
else{
echo "Nowa książka nie została dodana";
}
mysqli_close($link);
?>
```

Aktualizowanie danych

W celu zaktualizowania danych należy wykonać kilka czynności.

* Wyszukać dane do aktualizacji (jeden rekord)
* Wyświetlić je na formularzu
* Pobrać zaktualizowane w formularzu dane i zapisać je w bazie

Aktualizacja

```
<?php
$link = mysqli_connect("localhost", "root", "")
   or die("brak połączenia z serwerem");
mysqli_select_db($link,"henryks")
   or die("błąd bazy danych");
$query  = "SELECT * FROM lektury WHERE autor ='Adler E.' and tytul='Leonia'";
$zapytanie = mysqli_query($link,$query)
   or die("brak tabeli");
$ile=mysqli_num_rows($zapytanie);
if(!$zapytanie==true){
                        mysql_close($link);
                        exit( "Błąd w zapytaniu");
                                        }
                    if($ile==0)
                                                exit("Brak danych");
                    if($ile>1)
                                            exit("Nie można wyodrębnić danych");
$wiersz=mysqli_fetch_array($zapytanie);
$nazwisko=$wiersz['autor'];
$tytul=$wiersz['tytul'];
$numer=$wiersz['wydanie'];
$miejsce=$wiersz['miejsce'];
```

cd....

```
$miejsce=$wiersz['miejsce'];
$rok=$wiersz['rok'];
$cena=$wiersz['cena'];
$id=$wiersz['nr']     ?>
<!DOCTYPE HTML>
<html lang="pl">     <head>     <title>Aktualizacja książek</title>
<META HTTP-EQUIV="content-type" CONTENT="text/html; charset=UTF-8"></head><body>
    <form action="http://localhost/aktualizuj.php" method="post">
            <p><b>Modyfikacja</b></p><br/>
Nazwisko i imię autora:<br/><input tape="text" name="nazwisko" value="<?php echo
$nazwisko; ?>" size="30"><br/>
Tytuł:<br/>  <input tape="text" name="tytul" value="<?php echo $tytul; ?>" size="30"><br/>
Numer wydania:<br/> <input tape="text" name="numer" value="<?php echo $numer; ?>"
size="30"><br/>
Miejsce wydania<br/> <input tape="text" name="miejsce" value="<?php echo $miejsce; ?>"
size="30"><br/>
Rok wydania<br/> <input tape="text" name="rok" value="<?php echo $rok; ?>"
size="4"><br/> Cena:<br/>  <input tape="text" name="cena" value="<?php echo $cena; ?>"
size="30"><br/>
 Identyfikator:<br/> <input tape="text" name="id" value="<?php echo $id; ?>"
size="10"><br/>
<p><input type="submit" value="Aktualizuj" name="akt"></form></body></html>
```

Skrypt aktualizuj.php

Po połączeniu z serwerem i wybraniu bazy danych wywołana jest funkcja *mysqli_query()*, która wysyła zapytanie (*SELECT * FROM lektury WHERE autor=……*). W wyniku zostaną zwrócone dane wybranej pozycji. Funkcja *mysql_num_rows()* zwróci ilość znalezionych wierszy. Jeżeli wartość będzie większa od 1, to nie można określić jakie dane powinny być aktualizowane.

Jeżeli zostanie odczytana wybrana pozycja i dane zostaną wyświetlone na formularzu. Użytkownik może zaktualizować

aktualizuj.php

```
<?php
$link = mysqli_connect("localhost", "root", "")
   or die("brak połączenia z serwerem");

mysqli_select_db($link,"henryks")
   or die("błąd bazy danych");

$aktualny="UPDATE lektury SET
autor='$_POST[nazwisko]',tytul='$_POST[tytul]',wydanie='$_POST[numer]',miejs
ce='$_POST[miejsce]',rok='$_POST[rok]',cena='$_POST[cena]',nr='$_POST[id]'
WHERE autor='Adler E.' and tytul='Leonia'";
$zapytanie=mysqli_query($link,$aktualny);
if($zapytanie==true){
            echo "Dane zostały zaktualizowane";}
else{
            echo "Błąd, dane nie zostały zaktualizowane";}
mysqli_close($link);
?>
```

Tworzenie bazy danych

Do tworzenia bazy danych jest używana instrukcja SQL CREATE DATABASE.

Należy dołączyć ją do funkcji *mysql_guery().*

```
$link = mysqli_connect("localhost", "root", "")
   or die("brak połączenia z serwerem");

$baza="CREATE DATABASE Moje_filmy";
if(mysqli_query($link,$baza)){
            echo "Baza została utworzona";
}
else{
            echo "Błąd, Nie można utworzyć bazy";
}
?>
```

Tworzenie tabeli

Do tworzenia tabeli w istniejącej bazie danych jest używana instrukcja SQL

CREATE TABLE

```
$link = mysqli_connect("localhost", "root", "")
   or die("brak połączenia z serwerem");
mysqli_select_db($link,"moje_filmy")
   or die("błąd bazy danych");
$tab="CREATE TABLE wideo(nr INT,tytul TEXT,  aktor TEXT,dystrybucja TEXT,
            rok INT,   gatunek TEXT,cena INT) DEFAULT CHARACTER SET utf8";

$zapytanie=mysqli_query($link,$tab);
if($zapytanie==true)
{echo "Tabela została utworzona";}
else{
echo "Błąd, Nie można utworzyć tabeli";
}
```

Import danych cd..

```
$wstaw="INSERT INTO wideo(nr,tytul,aktor,dystrybucja,rok,gatunek,cena) VALUES
('1','Wschodni wiatr','Anderson','MMA','1977','Komedia','19'),
('2','Pułapka','Langer','Universe','1991','Horror','29'),
('3','Echa muzyki','Anderson','Wolf','1975','Muzyczny','27'),
.
.
('50','Sabrina','Boghart','Neilson','1954','Komedia','13')";
$pytanie=mysqli_query($link,$wstaw);
if($pytanie==true){
        echo "Tabela Wideo została zaimportowana";
}
else{
        echo "Błąd, Nie można zaimportować tabeli";
}
```

Uzupełnij import danych o pozostałe filmy z listy udostępnionej w pomocach: wideo.txt. Powinieneś mieć 50 pozycji w tabeli wideo.

Wykonaj samodzielnie

Na podstawie zdobytych wiadomości:

* Dodaj możliwość przeglądania filmów z Twojej filmoteki
* Wprowadź moduł dodawania nowych filmów
* Uzupełnij mini portal o możliwość aktualizowania danych
* Dla ambitnych: Może wyszukiwanie wg zadanych kryteriów ?

Biblioteka PDO (*PHP Data Objects*)

Rozszerzenie języka PHP udostępniające jednolity uniwersalny interfejs do komunikacji z bazami danych w technice obiektowej.

Zaletą PDO jest to, że tworzone skrypty mogą korzystać z tych samych metod, niezależnie od wykorzystywanej bazy. Kolejną zaletą jest walidacja danych wysyłanych w zapytaniach oraz filtrowanie tych danych pod kątem zabezpieczeń przed atakami.

Dostępne są sterowniki miedzy innymi do baz: MySQL, PostgreSQL, SQLite, Oracle, MS SQL/Sybase.

Nawiązanie połączenia

Aby nawiązać połączenie z bazą danych, należy za pomocą konstruktora utworzyć nowy obiekt klasy PDO w postaci:

PDO(źródło_danych, nazwa_użytkownika, hasło, opcje)

* źródło_danych-inaczej DSN(ang. Data Source Name), jest to specjalny ciąg znaków opisujący rodzaj bazy danych i sposób połączenia.
* opcje – to tablica zawierająca dodatkowe opcje związane z połączeniem.

Wersja uproszczona źródła danych dla MySQL wygląda następująco:

mysql:host=nazwa_serwera;port=numer_portu;dbname=nazwa_bazy

Przykład nawiązania połączenia

```
<?php
$user="janek";
$pass="haslo";
$dsn="mysql:host=localhost;dbname=lektury";
try{
        $pdo=new PDO($dsn,$user,$pass);
        echo 'Połączenie nawiązane!';}
catch(PDOException $e){
        echo 'Błąd połączenia:'.$e->getMessage();
        exit;}
?>
```

Utworzony **obiekt klasy PDO** został przypisany do zmiennej $pdo. Jeżeli podczas nawiązania połączenia wystąpi błąd, zostanie zgłoszony wyjątek PDOException. Musi on zostać przechwycony, ponieważ domyślny komunikat o błędzie, który wygeneruje PHP, wyświetli nazwę użytkownika i hasło do bazy danych.

Pobieranie danych

Zapytanie pobierające dane jest wysyłane za pomocą metody query(). Treść zapytania jest przekazywana do tej metody w postaci argumentu.

```
$obiekt_PDO->query(„Treść_zapytania”)
```

Metoda *query()* zwraca obiekt klasy *PDOStatement*, który zawiera wynik wykonywania zapytania.

Przykład pobierania danych

```
<?php
try{
        $pdo=new PDO('mysql:host=localhost;dbname=poligraf','root');
        $zapytanie=$pdo->query('SELECT autor, tytul FROM lektury');
        foreach($zapytanie as $wiersz){
                echo $wiersz['autor'].' , '.$wiersz['tytul'].'<br/>';}
        $pdo=null;}
catch(PDOException $e){
        echo "Błąd połączenia".$e->getMessage();}
?>
```

Metoda *query()* zwraca wynik wykonania zapytania. Za pomocą pętli *foreach* kolejne wiersze wyniku zostają zapisane do tablicy asocjacyjnej $wiersz i są wświetlane instrukcją echo. Połączenie zamykane jest poleceniem *$pdo=null.*

Aktualizacja danych

Zapytania aktualizujące typu INSERT, czy UPDATE są wysyłane za pomocą metody exec(). Po wykonaniu zapytania metoda zwraca liczbę określającą ilość zmodyfikowanych wierszy.

```
<?php
try{
        $pdo=new PDO('mysql:host=localhost;dbname=poligraf','root');
        $dodaj=$pdo->exec("INSERT INTO lektury(autor, tytul)
VALUES('Sienkiewicz','Trylogia')");
        if($dodaj>0){
        echo "Dodano".$dodaj."wierszy";}
        else{
                echo "Błąd podczas dodawania danych";}
        $pdo=null;
}
catch(PDOException $e){
        echo "Błąd połączenia".$e->getMessage();}
?>
```

Koniec

Czas na pytania kontrolne

1. Wymień obszary zastosowań języka PHP
2. W jaki sposób w kodzie HTML oznaczane są skrypty PHP?
3. Jakiego rodzaju informacje przechowują zmienne predefiniowane?
4. Czym różni się operator inkrementacji ++$x od operatora $x++?
5. Jakie jest przeznaczenie „*stałych magicznych*”
6. Gdzie stosowana jest pętla *foreach*?
7. Czym różnią się zmienne statyczne od zmiennych lokalnych?
8. Jaką wartość w języku PHP przyjmuje znacznik czasu(timestamp)?
9. Podaj definicję klasy w programowaniu obiektowym
10. Co zawierają składowe klasy?
11. Podaj przykład tworzenia obiektu w języku PHP
12. Na czym polega hermetyzacja w programowaniu obiektowym
13. Czym jest konstruktor w programowaniu obiektowym
14. Jakie zadanie w programowaniu obiektowym wykonuje destruktor?
15. Na czym polega dziedziczenie w programowaniu obiektowym
16. Na czym polega obsługa wyjątków w PHP
17. Czym różni się przesyłanie danych z formularza na serwer za pomocą metody POST od GET?
18. Jakie jest przeznaczenie plików cookies?
19. Jak działa dostępny w języku PHP mechanizm sesji?
20. Jakie parametry należy podać w kodzie PHP, aby nawiązać połączenie z bazą danych?
21. Jakie jest przeznaczenie biblioteki PDO dostępnej w języku PHP?

Praktyka

1. W MySQL utwórz bazę danych „Moja_szkoła”. Baza powinna zawierać tabelę **uczeń** z danymi uczniów oraz tabelę Rejestracja zawierającą dwa pola***:login i hasło***. Zostanie ona wykorzystana do rejestracji loginów i haseł użytkowników strony.
2. Utwórz stronę internetową, która za pomocą skryptów PHP będzie komunikowała się z tą bazą danych. Strona główna ma zawierać formularz logowania oraz formularz rejestracji nowych uczniów.
3. Utwórz skrypt który:
   * Dane z formularza rejestracyjnego zapisze w tabeli Uczeń
   * Login i hasło zapisze w tabeli rejestracja
   * Po zalogowaniu przekieruje użytkownika do strony umożliwiającej mu zamieszczanie komentarza
   * Wyświetli na tej stronie dotychczasowe wpisy użytkowników
   * Wyświetli nowy komentarz poniżej wcześniejszych komentarzy
   * Wyświetli na stronie przycisk wylogowania, a po kliknięciu wyloguje użytkownika ze strony