PHP – interpretowany, skryptowy język programowania zaprojektowany do generowania stron internetowych i budowania aplikacji webowych w czasie rzeczywistym.

uzycie php powoduje dynamiczne generwowanie kodu strony internetowej na serwerze

zalety php

\-wydajnos

\-skalowalnosc

\-interfesjy

Instrukcje echo print

<?php

//kropka “.” jest operatorem konkatencji

// czyli łączenie stringów

echo “<p>teskt1”.”<p>tekst2”;

print “<p>teskt1”.”<p>tekst2”;

?>

identyfikator jest nazwą utworzoną przez programiste moze on byc nazwa zmiennej lub funkcji podstawowe zasady tworzenia identyfikatorów:

-dowolona dlugosc

-dopuszczalne sa litery z kodu ASCII

-nie mozna rozpoczynac sie od cyfry

-rozroznienie sa wielkosci liter

-nazwa powinna jednoznaczenie wskazywac przeznaczenie

-przy dlugich nazwach stosowac tzw camel case

skrypty php mozna umieszczac w dowolnym miejscu pliku

w kodzie php mozna umiescic zwykly kod html, ktory zostanie przepisany do pliku wyjsciowego bez zmian

T: Typy danych

1. Typ danych określa rodzaj informacji, liczba, tekst. W jezyku php nie ma ścisłej kontrli typów gettype()- do sprawdzenia informacji sluzy ta funkcja. Szczegolwe informacje o zmiennej wypisze funkcja var\_dump(zmienna)
2. Typy danych

-bool-zmienna logiczna true/false

-int-liczby calkowite

-float-liczby zmienno przecinkowe

-string-ciag znaków

3.<?php

echo 'this is a simple string';

?>

do generowania niektorych znaków uzywamy tak zwanych znakow modyfikacj lub ucieczki i umieszczamy je w cudzyslowiach “ “

stringi w apostrofach sa interpretowane a w cudzyslowie cytowane

echo ‘this will not expand \n a new line’;

echo “thiss will expand”;

do sprawdzenia dlugosci stringa uzywamy funkcji

strien(),mb\_strien()

str replace() replace text within string

zamina tekstu

* \n - nowa linia
* \r - carriage return (resetowanie pozycji do początku wiersza)
* \tab - ab
* \\ - backslash
* \$ - dolar
* \’ - cudzysłów.

Znaki modyfikacji umieszczone w apostrofach są ignorowane, natomiast w cudzysłowach są interpretowane.

echo 'This is a simple string: \n a newline';

echo "This is a simple string: \n a newline";

Stringi w cudzysłowach interpretują zmienne:

$number=5;

echo "Variable value: $number";

Do sprawdzenia długości stringa używamy funkcji **strlen()** i **mb\_strlen()** - druga funkcja prawidłowo współpracuje z Polskimi literami

echo strlen("Łódź");

echo mb\_strlen("Łódź");

echo str\_word\_count("Hello world!")

## 

Zmienna jest elementem programu, który służy do przechowywania informacji. Nazwa zmiennej w PHP jest poprzedzona znakiem **$**

**$var = 'Bob';**

**$Var = 'Joe';**

**echo "$var, $Var";**

Zwykle zmiana wartości zmiennej następuje przez przypisanie

$foo = 'Bob'

Referencja jest to wskazanie innej zmiennej. Robimy to operatorem **ampersand &.**

$foo = 'Bob';

$bar = &$foo;

$bar = 'Andy';

echo $bar;

echo $foo;

Zmienne predefinowane:

**$GLOBALS** - zmienne o zasięgu globalnym,

**$\_SERVER** - zmienne tworzone przez serwer,

**$\_GET** - Przykładowo wpisanie w przeglądarce: ‘<http://www.mojadomena.org?id=5&kolor='czerwony>' przekaże do skryptu tablicę $\_GET z dwoma indeksami: $\_GET["id"] o wartości 5 oraz $\_GET["kolor"] o wartości 'czerwony',

**$\_POST** - wykorzystujemy do "obróbki" danych wprowadzanych za pomocą formularzy,

**$\_FILES** - Tablica asocjacyjna zmiennych zawierających informację o plikach wysłanych do serwera (uploadowanych) metodą POST,

**$\_REQUEST** - Tablica asocjacyjna, stanowiąca połączenie zmiennych z GET, POST i ciasteczek. Innymi słowy - cała informacja przychodząca od użytkownika,

**$\_SESSION** - Tablica przechowuje zmienne aktualnie zarejestrowane jako sesyjne,

**$\_ENV** - zmienne środowiskowe,

**$\_COOKIE** - ciasteczka,

**$php\_errormsg** - Poprzedni komunikat o błędzie,

**$http\_response\_header** - responsywne nagłówki HTTP,

**$argc** - Liczba argumentów przekazanych do skryptu (jeśli uruchamiany z linii poleceń),

**$argv** - Tablica argumentów przekazanych do skryptu.

**Zmienna statyczna** jest inicjowana tylko przy pierwszym uruchomieniu funkcji. Jest wykorzystywana jako licznik uruchomienia funkcji.

function test(){

static $licznik = 0;

echo $licznik . "<br>";

$licznik++;

}

test();

test();

test();

## 

1. Tablica jest zmienna która przechowuje wiele informacji umieszczonych jedna za drugą Numerowanie elementów (indeksowanie) rozpoczyna się od **zera.**
2. **Tablice indeksowane**
   1. tworzenie przy użyciu funkcji **array**

$cars = array("Volvo", "BMW", "Toyota");

* 1. skrócony sposób robienia tablicy

$cars = ["Volvo", "BMW", "Toyota"]

* 1. tworzenie pustej tablicy i dodawanie elementów ręcznie

$cars = [];

$cars[0] = "Volvo";

$cars[1] = "BMW";

$cars[2] = "Toyota";

* 1. Wywoływanie z tabeli

$books = array("pierwsza książka",

"drugi tytuł",

"trzecia książka");

echo $books[0];

echo $books[1];

* 1. Nie scalone teksty

$books = array("pierwsza książka",

"drugi tytuł",

"trzecia książka");

echo $books[0]. "<br>";

echo $books[1];

* 1. Nie scalone tekst (w źródłach strony)

$books = array("pierwsza książka",

"drugi tytuł",

"trzecia książka");

echo $books[0]. "<br>" . "\n";

echo $books[1];

1. Tablice asocjacyjneprzechowują informację parami
2. Tworzenie przy użyciu funkcji **array**

$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");

1. Krótka składnia

$age = ["Peter"=>"35", "Ben"=>"37", "Joe"=>"43"];

1. Dodawanie manualnie

$age = [];

$age = ['Peter'] = "35";

$age = ['Ben'] = "37";

$age = ['Joe'] = "43";

4. Tablice wielowymiarowejest to tablica, w której elementy są **tablicami**

$cars = array (

array("Volvo",22),

array("BMW", 15)

)

* 1. Tablice wielowymiarowe z użyciem **ECHO**

$cars = array (

array("Volvo", 22),

array("BMW", 15)

);

echo "<table>\n";

echo "<tr><td>Car</td><td>In stock</td></tr>\n";

echo "<tr><td>".$cars[0][0]."</td><td>".$cars[0][1]."</td></tr>\n";

echo "<tr><td>".$cars[1][0]."</td><td>".$cars[1][1]."</td></tr>\n";

echo "</table>"

* 1. Zwracanie liczby elementów //3 - w tym przypadku

$cars = array("Volvo", "BMW", "Toyota");

echo "Size of array cars = ". count($cars);

* 1. Sortowanie tablic indeksowanych

//bez wywołania

$cars = array("Volvo", "BMW", "Toyota");

sort($cars);

rsort($cars);

//z wywołaniem

$cars = array("Volvo", "BMW", "Toyota");

sort($cars);

var\_dump($cars);

rsort($cars);

var\_dump($cars);

* 1. Sortowanie tablic asocjacyjnych

$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");

asort($age);

ksort($age);

* 1. Usuwanie ostatniego elementu z **array**

$a = array("red", "green", "blue");

array\_pop($a);

print\_r($a);

* 1. Dodawanie jednego lub dwóch elementów na końcu **array**

$a = array("red", "green");

array\_push($a, "blue", "yellow");

print\_r($a);

* 1. Usuwanie pierwszego elementu z **array**

$a = array ("a"=>"red","b"=>"green","c"=>"blue");

echo array\_shift($a) . "<br>";

print\_r ($a);

* 1. Dodawanie pierwszego elementu dla **array**

$a = array ("c"=>"blue");

$n = array ("a"=>"red", "b"=>"green");

array\_unshift($a, $n);

print\_r ($a);

## 

1. Operatory są symbolami przedstawiającymi operacje na danych.
2. Operatory przypisania:
   1. Referencja jest to wskazanie na inną zmienna

$a = 3;

$b = &$a;

print "$a\n";

print "$b\n";

$a = 4;

print "$a\n";

print "$b\n";

* 1. Złożone operatory przypisania

$a += $b równoznaczne z $a = $a + $b

$a -= $b równoznaczne z $a = $a - $b

$a \*= $b równoznaczne z $a = $a \* $b

$a /= $b równoznaczne z $a = $a / $b

$a %= $b równoznaczne z $a = $a % $b

* 1. Operatory arytmetyczne

$a = 5;

$b = 3;

echo "liczby a=$a, liczba b=$b <br>";

echo "suma a+b=" . ($a + $b) . "<br>";

echo "różnica a-b=" . ($a - $b) . "<br>";

echo "iloczyn a\*b=" . ($a \* $b) . "<br>";

echo "iloraz a/b=" . ($a / $b) . "<br>";

echo "modulo a%b= " . ($a % $b) . "<br>";

echo "a do potęgi b = " . $a\*\*$b;

* 1. Inkrementacja/Dekrementacja operatorów

echo "<h3>Postincrement</h3>";

$a = 5;

echo "Should be 5: " . $a++ . "<br />\n";

echo "Should be 6: " . $a . "<br />\n";

echo "<h3>Preincrement</h3>";

$a = 5;

echo "Should be 6: " . ++$a . "<br />\n";

echo "Should be 6: " . $a . "<br />\n";

## 

1. Pętla służy do wielokrotnego wykonywania instrukcji
   1. Pętla **while**

**$x = 1;**

**while ($x <= 5){**

**echo "The number is: $x <br>";**

**$x++;**

**}**

* 1. Pętla **do…while** - instrukcje w pętli wykonają się przynajmniej jeden raz, ponieważ sprawdzenie warunku jest na końcu pętli

$x = 1;

do {

echo "The number is: $x <br>";

$x++;

} while ($x <= 5);

echo "<table border='1'>";

echo "<tr><th>Numer</th></tr>";

$x = 1;

do {

echo "<tr><td>Numer to: $x</td></tr>";

$x++;

} while ($x <= 5);

echo "</table>";

* 1. Pętla **for**

for ($x = 0; $x <= 5; $x++) {

echo "The number is: $x <br>";

}

* 1. Pętla **foreach**

$colors = array("red", "green", "blue", "yellow");

foreach ($colors as $value) {

echo "$value <br>";

}

$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");

foreach ($age as $key => $val) {

echo "$key = $val <br>";

}

Napisz skrypt który stworzy słownik owoców, napojów (cena, napoje) - tablica asocjacyjna w pętli)

$napoje = array("Piwko"=>"35,99", "Soczek"=>"37,99", "Pepsi"=>"43");

foreach ($napoje as $key => $val) {

echo "$key => $val zł <br>";

}

Teraz napisz to w tabeli

$napoje = array("Piwko"=>"35,99", "Soczek"=>"37,99", "Pepsi"=>"43");

echo '<table border="1">';

echo '<tr><th>Napój</th><th>Cena</th></tr>';

foreach ($napoje as $key => $val) {

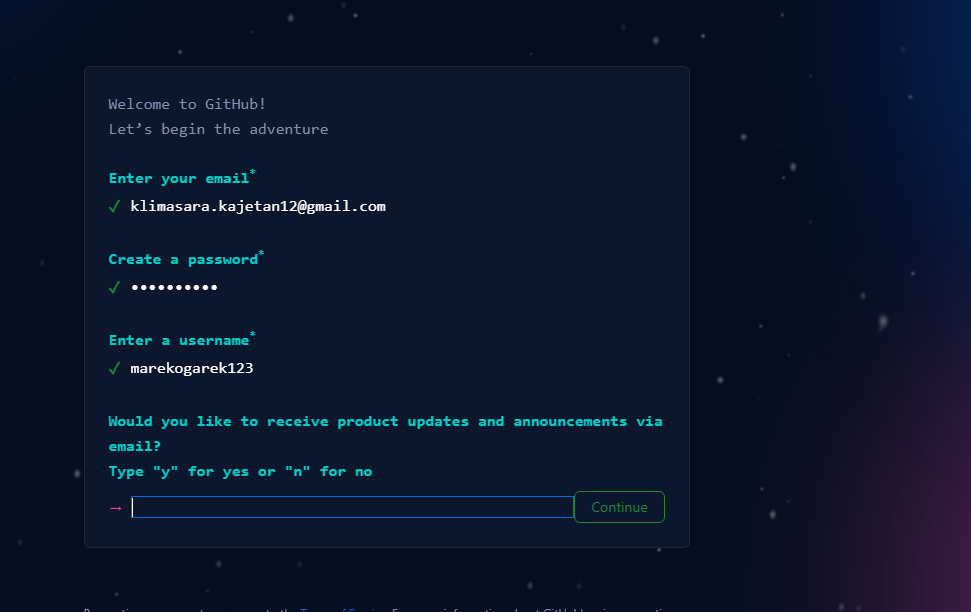
echo '<tr><td>' . $key . '</td><td>' . $val . ' zł</td></tr>';

}

echo '</table>';

## 

1. GIT jest to rozproszony system kontroli wersji.
2. System GIT stworzył **Linus Torvalds** (był twórcą linuxa).
3. Git jest dostarczany przez firm: **Bitbucket, GitHub, GitLab**.
4. Środowisko lokalne GITa pobieramy z [**https://git-scm.com/downloads**](https://git-scm.com/downloads)**.**
5. Zakładamy konto na GitHub.com



Funkcja to fragment programu, która realizuje określone zadanie

function writeMsg(){

echo "Hello world";

}

writeMsg();

Argumenty funkcji

Parametr formalny jest to nazwa zmiennej w funkcji. Parametr aktualny jest to wartość w wywołaniu funkcji

function familyName($fname){

echo "$fname Smith <br>";

}

familyName("John");

familyName("Jerry");

familyName("Tom");

Jeżeli do funkcji przekażemy referencje do zmiennej to możemy zmienić wartość tej zmiennej w miejscu wywołania

function add(&$string){

$string .= 'add some extra.';

}

$str = 'This is a string ';

add($str);

echo $str;

Instrukcja return służy do zwracania wartości

function square($num){

return $num \* $num;

}

echo square(4);asd

Funkcje anonimowe

Nie posiadają nazwy, są krótkie i są używane w miejscu gdzie powinno być wywołanie funkcji

1. Sprawdzenie, czy plik istnieje

$filename = '/path/to/foo.txt';

if (file\_exists($filename)) {

echo "The file $filename exists";

} else {

echo "The file $filename does not exist";

}

1. Odczyt wiersza

$myfile = fopen("webdictionary.txt", "r") or die ("Unable to open file!");

echo fgets($myfile);

fclose($myfile);

1. Otwarcie pliku

$myfile = fopen("newfile.txt", "w") or die ("Unable to open file!");

$txt = "John Doe\n";

fwrite($myfile, $txt);

fclose($myfile);

1. Data i czas

echo "Today is " . date("Y/m/d") . "<br>";

echo "Today is " . date("d.m.Y") . "<br>";

echo "Today is " . date("d-m-Y") . "<br>";

echo "Today is " . date("l") . "<br>";

1. Prosty zegar

echo "The time is " . date("h:i:sa");

**Data i czas**  w systemach komputerowych jest pamiętana jako liczba sekund od 1 stycznia 1970 roku. Znacznik czasu (ang. timestamp) jest to liczba sekund od 1 stycznia 1970.

$czas = mktime(11,14,54,8,12,2014);

echo "Created date is " . date( "d.m.Y h:i:sa", $czas);

$now = time();

$week = 7\*24\*60\*60;

echo "<br> teraz: ".date("d.m.Y h:i:sa", $now +$week);

1. Matura - Studniówka

$czas = mktime(11,14,54,5,6,2024);

echo "Matura " . date( "d.m.Y h:i:sa", $czas);

$now = time();

$week = 100\*24\*60\*60;

echo "<br> Studniówka: ".date("d.m.Y h:i:sa", $czas - $week);

## 

**Sesja** jest stanem podczas korzystania ze strony **WWW.** Dzięki niej serwer pamięta autoryzację użytkownika. Pierwszą czynnością w skrypcie jest uruchomienie funkcji **session\_start()** dzięki temu będziemy mieli dostęp do super globalnej tablicy **$session,** w której będą przechowywane zmienne sesji dostępne na innych podstronach.

Strona 1. Sesji

session\_start();

echo "Welcome to page #1";

$\_SESSION['favcolor'] = "blue";

$\_SESSION['animal'] = "dog";

$\_SESSION['time'] = time();

echo '<br /><a href="piatek-06-10(1).php">Page 2</a>';

Strona 2. Sesji

session\_start();

echo "Welcome to page #2<br>";

echo "Ulubiony kolor: ".$\_SESSION['favcolor']."<br>";;

echo "Ulubione zwierzę: ".$\_SESSION['animal']."<br>";

echo date('Y m d H:i:s', $\_SESSION['time'])."<br>";

echo $\_SESSION['bzdura'];

Zmienne sesji można usuwać zmienna **unset. unset ($\_SESSION[‘moja\_zmienna’]);.** Aby usunąć wszystkie zmienne sesji używamy funkcji **$\_SESSION = array().** Kiedy sesja wykonała już wszystkie działania, należy usunąć wszystkie zmienne, po czym wywołać: **session\_destroy();**

## 

Cookie są to informacje zapisywane przez serwer na komputerze lokalnym. Gdy przeglądarka łączy się z serwerem to najpierw sprawdza, czy są cookies dla tej strony, jeżeli tak to wysyła je do serwera.

$cookie\_name = "user";

$cookie\_value = "John Doe";

setcookie($cookie\_name, $cookie\_value, time() + (24\*60\*60));

Po utworzeniu Cookie zostaje ono zapamiętane w tablicy **$\_COOKIE.**

$cookie\_name = "user";

if(!isset($\_COOKIE[$cookie\_name])) {

echo "Cookie named '" . $cookie\_name . "' is not set!";

} else {

echo "Cookie '" . $cookie\_name . "' is set!<br>";

echo "Value is: " . $\_COOKIE[$cookie\_name];

}

Kasowanie Cookie - polega na przypisaniu mu daty ważności, która już minęła.

$cookie\_name = "user";

setcookie($cookie\_name, "", time() - 3600);

if(!isset($\_COOKIE[$cookie\_name])) {

echo "Cookie named '" . $cookie\_name . "' is not set!";

} else {

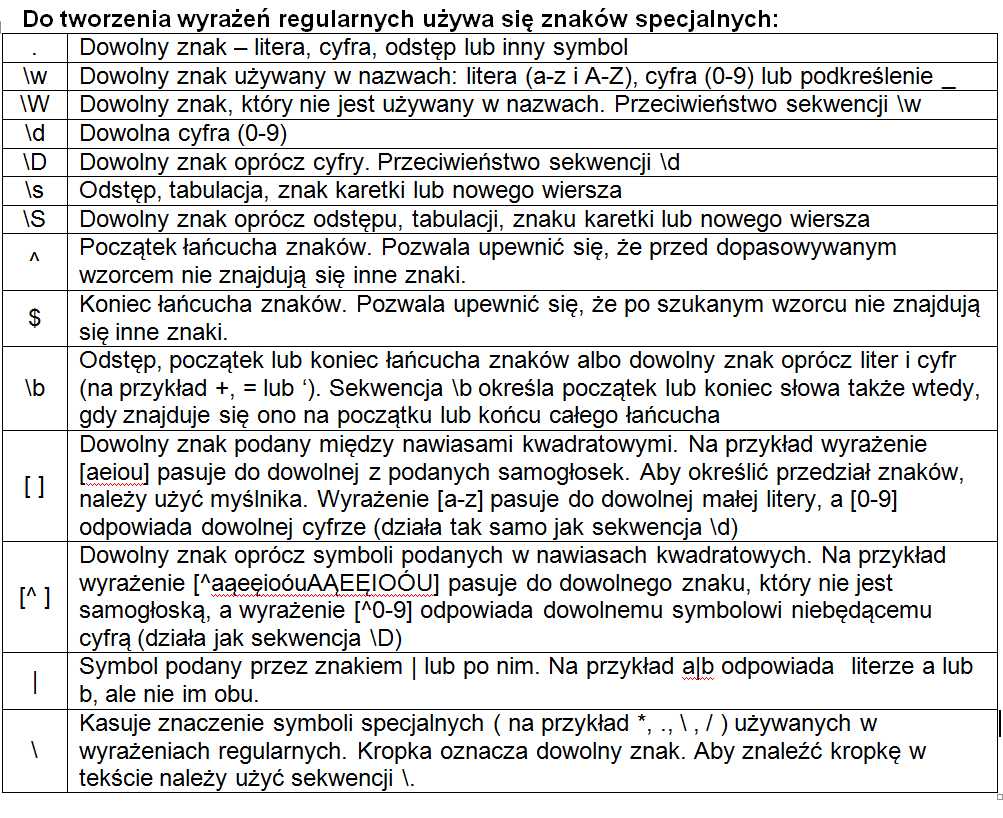
echo "Cookie '" . $cookie\_name . "' is set!<br>";

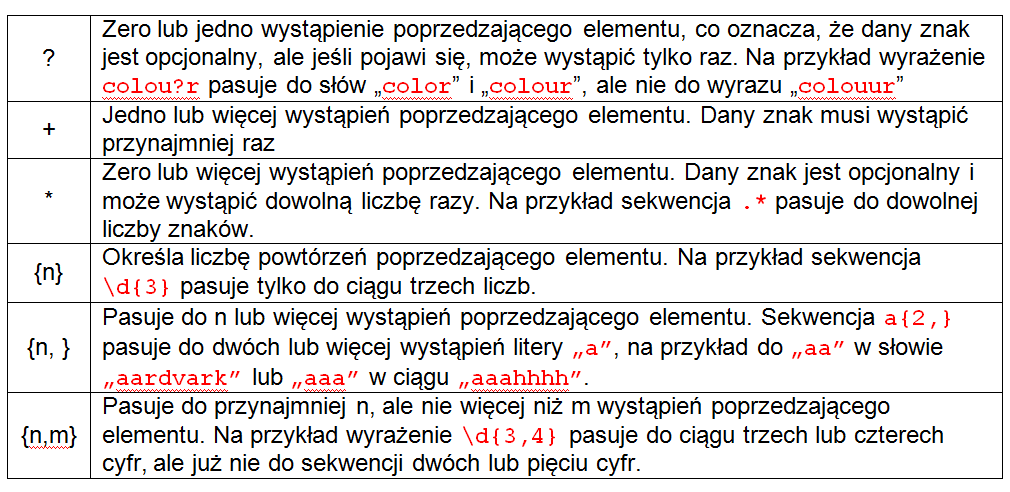
echo "Value is: " . $\_COOKIE[$cookie\_name];

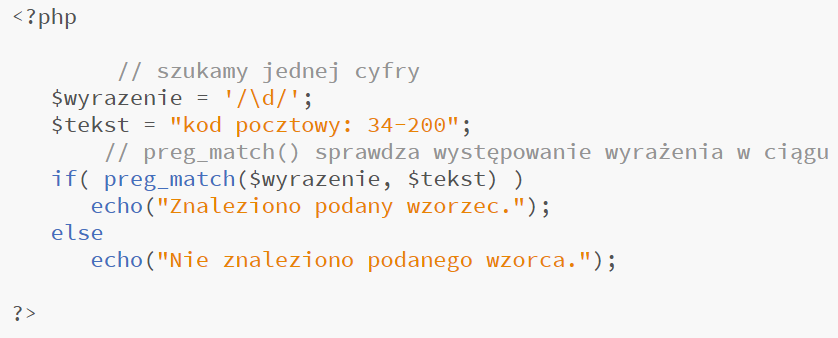
}

## 

1. Wyrażenia regularne służą do określenia wzorca, wyszukiwania informacji.







Wyszukiwanie cyfry:

$wzorzec = '/\d/';

$tekst = "mam 4 zeszyty";

if (preg\_match($wzorzec, $tekst))

echo "znaleziono cyfrę";

else

echo "nie znaleziono cyfry";

Wyszukiwanie pliku .gif

$wzorzec = '/\d{5}/';

$tekst = "telefon 564324453";

if (preg\_match($wzorzec, $tekst))

echo "znaleziono cyfrę";

else

echo "nie znaleziono cyfry";

Wyszukiwanie kodu pocztowego:

$wzorzec = '/\d{2}-\d{3}/';

$tekst = "poczta 34-200 Sucha Beskidzka";

if (preg\_match($wzorzec, $tekst, $matches))

echo "znaleziono kod pocztowy" . $matches[0];

else

echo "nie znaleziono kodu pocztowego";

Wyszukiwanie dat:

$wzorzec = '/([0123]?\d)+[-\.\/]+([01]?\d)+[-\.\/]+(\d{4})/';

**([0123]?\d)** - dzień - pierwsza cyfra może mieć wartość 0, 1, 2 lub 3 i jest opcjonalna druga cyfra jest dowolna,

**[-\.\/]** - separator - reprezentuje myślnik, ukośnik odwrócony lub kropkę rozdzielającą liczby dat,

**([01]?\d)** - miesiąc - pierwsza cyfra może być 0 lub 1 i jest opcjonalna, druga cyfra jest dowolna,

**(\d{4})** - rok - dokładne cztery cyfry roku

Wyszukiwanie daty:

$wzorzec = '/([0123]?\d)+[-\.\/]+([01]?\d)+[-\.\/]+(\d{4})/';

$tekst = "Sucha Beskidzka, dn 04.10.2023";

if (preg\_match($wzorzec, $tekst))

echo "znaleziono datę";

else

echo "nie znaleziono daty";

Wyszukiwanie adresu email:

$wzorzec = '/[-\w.]+@([-A-z0-9]+\.)+[A-z]/';

**[-\w.]** - myślnik, dowolny znak, kropka,

**+** -jedno lub więcej wystąpień,

**@** - znak używany w adresach,

**[A-z0-9]** - jedna litera lub cyfra,

**[-A-z0-9]+** - pasuje do przynajmniej jednego wystąpienia litery, cyfry lub myślnika,

**\.** - Oznacza kropkę,

**()+** - odpowiada przynajmniej jednemu wystąpieniu wzorca (umieszczone w nawiasie).

$wzorzec = '/[-\w.]+@([-A-z0-9]+\.)+[A-z]/';

$tekst = "email: john@poczta.eu";

if (preg\_match($wzorzec, $tekst))

echo "znaleziono email";

else

echo "nie znaleziono email";

Wyszukiwanie adresu WWW:

$wzorzec = '/((\bhttps?:\/\/)|(bwww\.))\S\*/';

**\bhttp** - adres może zaczynać się od http,

**s?** - może być w wersji https, dlatego s jest opcjonalne (s?),

**:** - zwykły znak,

**\/\/** - po frazie http: występują dwa ukośniki //, ale ukośnik oznacza koniec wyrażenia regularnego. Aby zapisać ukośnik we wzorcu należy poprzedzić go znakiem \,

**()|()** - Kreska pionowa oznacza, że może wystąpić jedno z wyrażeń umieszczonych między nawiasami,

**/bwww** - adres może zaczynać się od WWW,

**\.** - po WWW występuje kropka,

**\S\*** - dowolna liczba znaków różnych od odstępu.

$wzorzec = '/((\bhttps?:\/\/)|(bwww\.))\S\*/';

$tekst = "adres: www.zsgsucha.pl";

if (preg\_match($wzorzec, $tekst))

echo "znaleziono adres www";

else

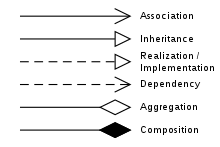
echo "nie znaleziono adresu www";

## 

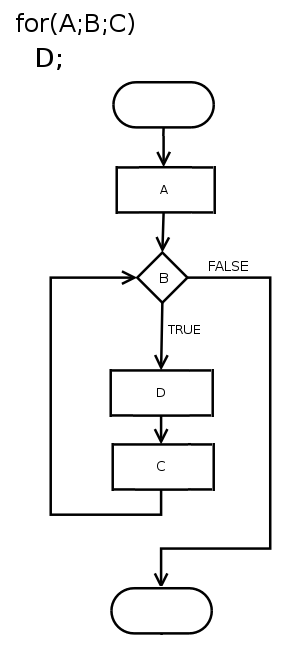
1. Programowanie strukturalneopiera się na podziale kodu źródłowego na procedury i bloki w postaci instrukcji wybór i pętli
2. Programowanie obiektowew którym programy definiuję się za pomocą obiektów - stan i zachowanie
3. Zalety programowania obiektowego:
   1. OOP jest szybsze i łatwiejsze do wykonania,
   2. Zapewnia prostą strukturę,
   3. Ułatwia stosowania metody DRY (Don't Repeat Yourself),
   4. Zwiększa bezpieczeństwo programu
4. **)** - Zunifikowany język modelowania.
5. Diagram klas przedstawia budowę obiektów



1. Widoczność atrybutów - dla atrybutów możemy zdefiniować dostępność:
   1. + - publiczny,
   2. # - chroniony,
   3. - - prywatny,
2. Zależności między klasami



1. Diagram przypadków użycia - przedstawia usługi które system świadczy aktorom
2. Diagram czynności - modeluje czynności i odpowiedzialność elementów



1. Diagram stanów - pokazuje możliwe stany obiektu oraz przejścia do tych stanów
2. Klasa - opisuje budowę obiektu

class Fruit {

public $color;

public $name;

function set\_name($name) {

$this -> name = $name;

}

function get\_name() {

return $this->name;

}

}

1. Tworzenie i używanie obiektów

$apple = new Fruit();

$banana = new Fruit();

$apple -> set\_name('Apple');

$banana -> set\_name('Banana');

echo $apple -> get\_name();

echo "<br>";

echo $banana -> get\_name();

1. Konstruktor - jest funkcją składową klasy o nazwie **\_\_construct.** Jest ona automatycznie uruchamiana przy tworzeniu obiektu. Dlatego najczęściej wykorzystuje się ją do inicjalizacji obiektu.

class Fruit {

public $color;

public $name;

function \_\_construct($name) {

$this -> name = $name;

}

function get\_name() {

return $this->name;

}

}

$apple=new Fruit("apple");

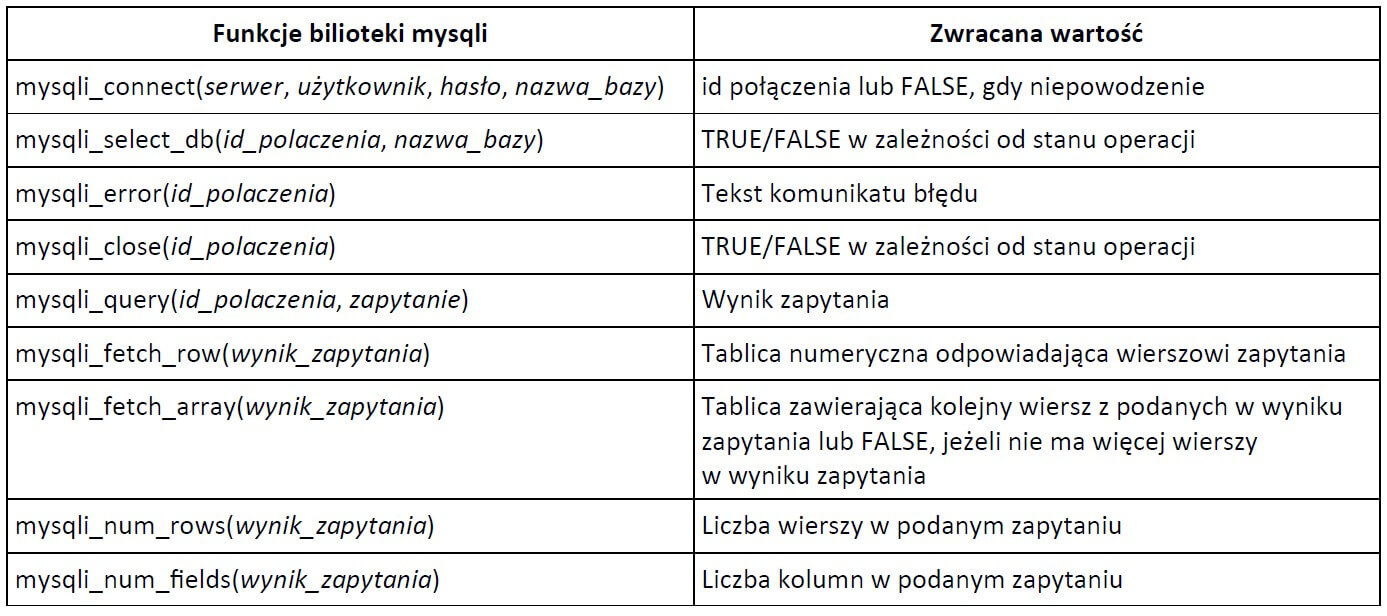
echo $apple->get\_name();

1. Destruktor - jest to składowa klasy o nazwie **\_\_destruct,** która jest automatycznie uruchomiona przy kasowaniu obiektu, służy ona do sprzątania po obiekcie

## 

**API** - Application Programming Interface - Funkcje do obsługi różnych zastosowań.

**PDO** - PHP Data Objects - jest to język PHP do komunikacji z bazą danych.



1. Połączenie z bazą danych

$servername = "localhost";

$username = "root";

$password = "";

$conn = mysqli\_connect($servername, $username, $password);

if (!$conn) {

die("Connection failed: ". mysqli\_connect\_error());

}

echo ("Connected successfully");

1. Stworzenie bazy danych

$database = "5ina\_1";

$sql = "CREATE DATABASE $database";

if (mysqli\_query($conn, $sql)) {

echo ("Database created successfully");

} else {

echo ("Error creating database: ". mysqli\_error($conn));

}

1. Połączenie z wybraną bazą

if(mysqli\_select\_db($conn, $database)) {

echo ("Database $database selected");

} else {

echo ("Error select database: " . mysqli\_error($conn));

}

1. Tworzenie tabeli

if(mysqli\_select\_db($conn, $database)) {

echo ("Database $database selected");

} else {

echo ("Error select database: " . mysqli\_error($conn));

}

$sql = "CREATE TABLE MyGuests (

id INT(6) UNSIGNED AUTO\_INCREMENT PRIMARY KEY,

firstname VARCHAR(30) NOT NULL,

lastname VARCHAR(30) NOT NULL,

email VARCHAR(30),

reg\_date TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP

)";

if (mysqli\_query($conn, $sql)) {

echo ("Table MyGuest created successfully");

} else {

echo ("Error creating table: " . mysqli\_error($conn));

}

1. Insert Data do Bazy danych

$sql = "INSERT INTO MyGuests (firstname, lastname, email)

VALUES ('John', 'Doe', 'john@example.com')";

if (mysqli\_query($conn, $sql)) {

echo "New record created successfully";

} else {

echo "Error" . $sql . "<br>" . mysqli\_error($conn);

}

1. Dodawanie wielokrotnych rekordów

$sql = "INSERT INTO MyGuests (firstname, lastname, email)

VALUES ('John', 'Doe', 'john@example.com');";

$sql .= "INSERT INTO MyGuests (firstname, lastname, email)

VALUES ('Mary', 'Moe', 'mary@example.com');";

$sql .= "INSERT INTO MyGuests (firstname, lastname, email)

VALUES ('Julie', 'Joe', 'julie@example.com');";

if(mysqli\_multi\_query($conn, $sql)) {

echo "New record created successfully";

} else {

echo "Error" . $sql . "<br>" . mysqli\_error($conn);

}

1. Select data

$sql = "SELECT id, firstname, lastname FROM MyGuests";

$result = mysqli\_query($conn, $sql);

if(mysqli\_num\_rows($result) > 0) {

while($row = mysqli\_fetch\_assoc($result)) {

echo "id: " . $row["id"] . " - Name" . $row["firstname"] . "" . $row["lastname"] . "<br>";

}

} else {

echo "0 results";

}

1. Select data v2

$sql = "SELECT id, firstname, lastname FROM MyGuests";

$result = mysqli\_query($conn, $sql);

if(mysqli\_num\_rows($result) > 0) {

while($row = mysqli\_fetch\_row($result)) {

echo "id: " . $row[0] . " - Name" . $row[1] . "" . $row[2] . "<br>";

}

} else {

echo "0 results";

}

1. Generowanie tabeli

$sql = "SELECT id, firstname, lastname FROM MyGuests";

$result = mysqli\_query($conn, $sql);

if(mysqli\_num\_rows($result) > 0) {

echo "<table>";

while($row = mysqli\_fetch\_row($result)) {

echo "<tr><td>" . $row[0] . "</td><td>" . $row[1] . "</td><td>" . $row[2] . "</td></tr>";

}

echo "</table>";

} else {

echo "0 results";

}

1. Generowanie listy

<?php

$sql="SELECT id, firstname, lastname FROM MyGuests";

$results=mysqli\_query($conn,$sql);

if (mysqli\_num\_rows($result)>0) {

echo "<ul>";

while($row=mysqli\_fetch\_row($result)) {

echo "<li>".$row[0]." ".

$row[1]." ".$row[2]."</li>";

}

echo "</ul>";

}else {

echo "0 results";

}

?>

1. Kasowanie danych

<?php

$sql = "DELETE FROM MyGuests WHERE id=3";

if (mysqli\_query($conn, $sql)){

echo "Record deleted successfully";

} else {

echo "Error deleting record: " . mysqli\_error($conn);

}

?>

1. Obsługa formularzy

Plik html:

<form action="Form\_post.php" method="post">

Name: <input type="text" name="name"><br>

E-mail: <input type="text" name="email"><br>

<input type="submit">

</form>

Plik php:

Welcome <?php echo $\_POST["name"]; ?><br>

Your email address is: <?php echo $\_POST["email"]; ?>

Radio button html:

<p>Choose your favorite Web language</p>

<form action="mysql.php" method="POST">

First name: <input type="text" name="firstname"><br>

Last name: <input type="text" name="lastname"><br>

<input type="radio" id="html" name="fav\_language" value="HTML">

<label for="html">HTML</label><br>

<input type="radio" id="css" name="'fav\_language" value="HTML">

<label for="css">CSS</label><br>

<input type="radio" id="javascript" name="fav\_language" value="HTML">

<label for="javascript">JavaScript</label><br>

<input type="submit">

</form>

Radio button PHP:

$firstname=$\_POST["firstname"];

$lastname=$\_POST["lastname"];

$fav\_language=$\_POST["fav\_language"];

$sql="INSERT INTO MyGuests (firstname, lastname, fav\_language) VALUES ('$firstname', '$lastname', '$fav\_language')";

if (mysqli\_query($conn, $sql)){

echo "New record created successfully";

} else {

echo "Error: " . $sql . "<br>" . mysqli\_error($conn);

}

Obsługa formularzy - checkbox

plik - formularz1.html:

<form action="checkbox\_post1.php" method="POST">

<input type="checkbox" id="vehicle1" name="vehicle1" value="Bike">

<label for="vehicle1">I have a bike</label><br>

<input type="checkbox" id="vehicle2" name="vehicle2" value="Car">

<label for="vehicle2">I have a car</label><br>

<input type="submit" value="Submit">

</form>

plik - checkbox\_post1.php:

echo "Pierwszy Checkbox: " . $\_POST["vehicle1"];

echo "Drugi Checkbox: " . $\_POST["vehicle2"];

## 

1)

<p>Form validation</p>

Name: <input type="text" name="Name"><br>

E-mail <input type="text" name="E-mail"><br>

Website <input type="text" name="Website"><br>

Comment <textarea type="text" name="comment"></textarea><br>

Gender: <input type="radio" id="femele" name="gender" value="female">

<label for="female">Female</label>

<input type="radio" id="male" name="gender" value="male">

<label for="male">Male</label>

<input type="radio" id="other" name="gender" value="other">

<label for="other">Other</label></br>

<input type="submit">

2)

<!DOCTYPE html>

<html lang="pl">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Document</title>

</head>

<style>

.error {

color: #FF0000;

}

</style>

<body>

<?php

$nameErr = $emailErr = $genderErr = $websiteErr = "";

$name = $email = $gender = $comment = $website = "";

?>

<form method="POST" action="<?php echo htmlspecialchars($\_SERVER["PHP\_SELF"]);?>">

Name: <input type="text" name="Name" value="<?php echo $name;?>"><span class="error">\* <?php echo $nameErr;?></span><br><br>

E-mail: <input type="text" name="E-mail" value="<?php echo $email;?>"><span class="error">\* <?php echo $emailErr;?></span><br><br>

Website: <input type="text" name="Website" value="<?php echo $website;?>"><span class="error">\* <?php echo $websiteErr;?></span><br><br>

Comment <textarea type="text" name="comment" rows="5" cols="40"><?php echo $comment;?></textarea><br><br>

Gender: <input type="radio" id="femele" name="gender" <?php if(isset($gender) && $gender == "female") echo "checked";?> value="female">Female

<input type="radio" id="femele" name="gender" <?php if(isset($gender) && $gender == "male") echo "checked";?> value="male">Male

<input type="radio" id="femele" name="gender" <?php if(isset($gender) && $gender == "other") echo "checked";?> value="other">other

<span class="error">\* <?php echo $genderErr;?></span>

<br><br>

<input type="submit" name="submit" value="Submit">

</form>

</body>

</html>

3)

if(empty($\_POST["name"])) {

$nameErr = "Name is required";

} else {

$name = test\_input($\_POST["name"]);

if (!preg\_match("/^[a-zA-Z-']\*$/", $name)) {

$nameErr = "Only letters and white space allowed";

}

}

if(empty($\_POST["email"])) {

$emailErr = "Email is required";

} else {

$email = test\_input($\_POST["email"]);

if (!filter\_var($email, FILTER\_VALIDATE\_EMAIL)) {

$emailErr = "Invalid email format";

}

}

if (empty($\_POST["website"])) {

$website = "";

} else {

$website = test\_input($\_POST["website"]);

if (!preg\_match("/\b(?:(?:https?|ftp):\/\/|www\.)[-a-z0-9+&@#\/%?=~\_|!:,.;]\*[-a-z0-9+&@#\/%=~\_|]/i", $website)) {

$websiteErr = "Invalid URL";

}

}

if (empty($\_POST["comment"])) {

$comment = "";

} else {

$comment = test\_input($\_POST["comment"]);

}

if (empty($\_POST["gender"])) {

$genderErr = "Gender is required";

} else {

$gender = test\_input($\_POST["gender"]);

}

function test\_input($data) {

$data = trim($data);

$data = stripslashes($data);

$data = htmlspecialchars($data);

return $data;

}