Podstawy PHP

Wprowadzenie

PHP jest językiem skryptowym działającym po stronie serwera, używa się go do tworzenia dynamicznych stron lub aplikacji internetowych. Kod PHP umieszcza się w plikach z rozszerzeniem ".php". Parser PHP podczas przetwarzania plików przesyła zawarty w nich kod HTML w niezmienionej formie do klienta (przeglądarki), jeśli napotka jeden ze znaczników otwarcia kodu PHP np.: "<?php" i zamknięcia "?>" to wykonuje instrukcje między nimi zawarte. Wynikiem tych instrukcji jest przeważnie kod HTML, który następnie jest również wysyłany do klienta.

Podstawowe instrukcje i funkcje PHP

Wszystkie instrukcje PHP umieszcza się między znacznikami <?php KOD PHP ?>.

Aby wyświetlić tekst należy użyć instrukcji echo:

```
echo '<h1>Ćwiczenia z podstaw PHP.</h1>';
```

Wyświetlenie 3 poziomów nagłówków:

```
for ($i = 1; $i < 4; $i++)
{
  echo "<h$i>Nagłówek z poziomu nr $i</h$i>";
}
```

Przykłady wyświetlania zmiennej wewnątrz tekstu:

```
$x = 4;
echo 'Zmienna $x = $x';      // Zmienna $x = $x
echo 'Zmienna $x = '.$x;     // Zmienna $x = 4
echo "Zmienna $x = $x";      // Zmienna 4 = 4
echo "Zmienna \$x = $x";     // Zmienna $x = 4
```

Poniżej przykład wyświetlenia listy nazwisk i imion w tabeli.

```
<style>
table { border-spacing: 0px; border-collapse: collapse; }
td { padding: 2px; border: solid 1px black; }
</style>

<?php
$lista = array('Jan Kowalski', 'Kasia Wola', 'Robert Kowal', 'Monika
Kulak', 'Marek Nowak');    // lista nazwisk

echo '<table style="border: solid 1px black">';
echo '<tr><td>Id</td><td>Imię</td><td>Nazwisko</td></tr>';

foreach($lista as $id => $element)
// $id to kolejny numer elementu w tablicy, a $element zawiera
wartość tego elementu.
{
```

```
  $wiersz = '<tr>';
  $wiersz .= '<td>' . ($id + 1) . '.</td>';
// Dodajemy liczbę 1 gdyż domyślnie tablica jest numerowana od zera.
  list($imie, $nazwisko) = explode(' ', $element);
  $wiersz .= "<td>$imie</td><td>$nazwisko</td>";
  $wiersz .= '</tr>';
  echo $wiersz;
}
echo "</table>";
?>
```

Powyższą pętle można zapisać także w inny sposób:

```
for($id = 0; $id < count($lista); $id++)
{
  $element = $lista[$id];
  //...
}
```

Użyta funkcja `explode` rozbija tekst `(string $wiersz)` według podanego ciągu znaków (`string`) w pierwszym parametrze `(' ')` i tworzy z tych części tablicę.

```
$tablica_kolorow = explode(',', 'czarny,czerwony,niebieski');
```

Taka tablica będzie zawierała 3 elementy: 'czarny', 'czerwony', 'niebieski'.

Funkcja `list(…)` nie jest zwykłą funkcją, jest to element składni języka PHP. Jeśli przypiszemy do niej tablicę to każdy element tej tablicy zostanie przypisany do odpowiedniej zmiennej, która jest parametrem tejże funkcji.

```
list($kolor_czarny, $kolor_czerwony, $kolor_niebieski) =
$tablica_kolorow;
```

Teraz można użyć nowych zmiennych, które odpowiednio będą zawierały kolejne wartości z tablicy.

```
echo $kolor_czarny;        // 'czarny'
echo $kolor_czerwony;      // 'czerwony'
echo $kolor_niebieski;     // 'niebieski'
```

Aby co druga linijka w powyższej tabelce była w innym kolorze należy trochę zmodyfikować kod PHP i CSS. Do stylów dodajemy dwa wpisy tworzące klasy dla elementu `<tr>`:

```
tr.kolor1 { background-color: #fffacd; }
tr.kolor2 { background-color: white; }
```

Modyfikujemy wnętrze pętli. Zamiast linii: `$wiersz = '<tr>';`
wpisujemy:

```
if ($id % 2 == 0) $nr = 1; else $nr = 2;
//można też tak: $nr = $id % 2 == 0 ? 1 : 2;
$wiersz = "<tr class=\"kolor$nr\">";
```

Dzięki temu jeśli zmienna `$id` będzie przechowywała wartość parzystą to element `<tr>` będzie miał klasę stylów ustawioną na `"klasa1"`, a w odwrotnym przypadku na `"klasa2"`. Powyższy kod można również zapisać w postaci:

```
$wiersz = '<tr class="kolor'.($id % 2 == 0 ? 1 : 2).'">';
```

lub

```
$wiersz = '<tr class="kolor'.(($id % 2) + 1).'">';
```

Często powtarzające się instrukcje w skrypcie powinny być zawarte w funkcjach:

```
function RokPrzestepny($rok)
{
  if (!is_int($rok)) $rok = (int)$rok;
  return ($rok % 4 == 0) ? 'tak' : 'nie';
}
$r = 2010;
$odp = RokPrzestepny($r);
echo "Rok $r (przestępny: $odp)<br />"; //Rok 2010 (przestępny: nie)
$r = 2008;
$odp = RokPrzestepny($r);
echo "Rok $r (przestępny: $odp)<br />"; //Rok 2008 (przestępny: tak)
```

Utworzona została funkcja sprawdzająca czy podany rok w parametrze funkcji `($rok)` jest rokiem przestępnym. Jeśli rok jest przestępny to funkcja zwraca tekst "tak", a w innym przypadku "nie". Instrukcja `return` służy do zwracania wartości funkcji. W powyższym przykładzie wykorzystania funkcji widać powtarzające się elementy, które mogą być również zawarte wewnątrz funkcji, dzięki czemu sześć linijek można zapisać w dwóch liniach kodu.

```
function WyswietlRok($rok)
{
  $tekst = "Rok [rok] (przestępny: [odp])<br />";
  $tab1 = array('[rok]', '[odp]');
  $tab2 = array($rok, RokPrzestepny($rok));
  return str_replace($tab1, $tab2, $tekst);
}
echo WyswietlRok(2010); //Rok 2010 (przestępny: nie)<br />
echo WyswietlRok(2008); //Rok 2008 (przestępny: tak)<br />
// itd. ...
```

Nie każda funkcja musi zwracać wartości. Poniżej przykład tej samej funkcji tylko, że tym razem funkcja nic nie zwraca.

```
function WyswietlRok($rok)
{
  $tekst = "Rok [rok] (przestępny: [odp])<br />";
  $tab1 = array('[rok]', '[odp]');
  $tab2 = array($rok, RokPrzestepny($rok));
  echo str_replace($tab1, $tab2, $tekst);
}
WyswietlRok(2010);         //Rok 2010 (przestępny: nie)<br />
WyswietlRok(2008);         //Rok 2008 (przestępny: tak)<br />
```

Funkcja `str_replace` służy do zamiany znaków w ciągu (`string`). Pierwszy parametr zawiera `string/tablicę string`, które mają być zastąpione. Drugi parametr zawiera `string/tablicę string`, które mają zastąpić parametr pierwszy. Ostatni parametr to `string` podlegający zamianie określonej przez parametry pierwszy oraz drugi.

```
echo str_replace(array('a', 'b'), array('A', 'B'), 'abcabc');
// ABcABc
echo str_replace(array('a', 'b', 'c'), array('A', 'B'), 'abcabc');
// ABAB - c zostało zastąpione "niczym" ;)
echo str_replace(array('a', 'b'), '.', 'abcabc');
// ..c..c
```

Innymi przydatnymi funkcjami operującymi na ciągach znaków są:
`str_len()` – funkcja zwracająca ilość znaków w ciągu `string`, np.:

```
echo str_len('abcd');      // 4
```

`substr()` - zwraca część stringa, np.:
```
echo substr('abcde', 1, 3);      // bcd
```

`trim()` - usuwa z początku i końca stringa znaki białe (spacje, entery, tabulatory, itd.), np.:
```
$tekst = trim('  abc def\n');    // $teskt = 'abc def'
```

`Strpos()` - zwraca pierwsze wystąpienie stringa, np:
```
$i = strpos('abc abc', 'a');          // $i = 0
$i = strpos('abc abc', 'a', true);     // $i = 4
```

Jeśli podana fraza nie zostanie znaleziona to funkcja zwróci `false/0` co może sugerować, że fraza znajduje się w elemencie nr 0 a to jest nieprawdą. Dlatego zawsze trzeba sprawdzić czy zwracany typ jest typu `int` czy `bool`. Można to zrobić na dwa sposoby:
```
if( !is_int($i) ) echo 'Nie znaleziono szukanej frazy';
```
lub
```
if( $i === false ) echo 'Nie znaleziono szukanej frazy';
```

Operator `===` sprawdza czy obie wartości są takie same i w odróżnieniu od operatora `==` dodatkowo sprawdza czy są tego samego typu.

Na stronie **www.php.net** można znaleźć wiele innych ciekawych i przydatnych funkcji.