Lekcja 1

Czym są skrypty PHP?

Skrypty PHP są to programy umieszczane w treści stron WWW. Są one wykonywane przez serwer przed wysłaniem strony do użytkownika. Mają bardzo duże możliwości, ale mimo to są dość proste. Bardzo łatwa jest obsługa formularzy za pomocą skryptów PHP. Jedną z największych zalet skryptów PHP jest bardzo łatwa integracja z wieloma bazami danych. Oprócz tego skrypty PHP mają wiele innych ciekawych możliwości, jak dynamiczne tworzenie obrazków w formacie GIF, i możliwość łatwego wysyłania plików na serwer. Dzięki temu skrypty PHP stanowią ciekawą alternatywę dla skryptów CGI pisanych w Perlu, czy innych językach.

Jak umieścić skrypt na stronie

Skrypt PHP można umieścić w pliku HTML o rozszerzeniu ustalonym przez administratora serwera. Zwykle są to pliki *.php3 dla wersji 3.0 i *.phtml dla wersji wcześniejszych. Fragment dokumentu który ma zastać zinterpretowany jako skrypt można zaznaczyć na 4 sposoby:
1. <? (treść skryptu) ?>
2. <?php (treść skryptu) ?>
3. <script language="php"> (treść skryptu) </script>
4. <% (treść skryptu) %> (od wersji 3.0.4)
Wybór użytego sposobu może zależeć od innych rozszerzeń języka HTML, ewentualnie od ograniczeń zastosowanego do tworzenia strony edytora (w tym wypadku zwykle używa się sposobu 3).

Najprostszy skrypt

Na początku jedna uwaga: znaczniki <? i ?> można zastąpić dowolnymi wymienionymi wyżej. W następnych przykładach będe je pomijał, podając jedynie treść skryptu.

```
<? echo("To jest prosty skrypt
PHP"); ?>
```

W miejscu gdzie zostanie wstawiony ten skrypt powinien pojawić się napis To jest prosty skrypt PHP. Ten skrypt składa się tylko z jednej instrukcji. Instrukcje w skryptach PHP są kończone średnikami. Znacznik końca skryptu także kończy instrukcję, więc średnika na końcu tej lini mogłoby nie być. Dobrym zwyczajem jest jednak stawianie średnika także po ostatniej instrukcji. Pomaga to w unikaniu pomyłek - jeżeli teraz dopisze jakieś instrukcje na końcu skryptu wszystko będzie w porządku. Gdyby średnika nie było, pojawiłby się błąd.

Komentarze

Komentarz to wstawiany w skrypcie kawałek tekstu który jest po prostu ignorowany. Komentarze są przydatne zwłaszcza przy pisaniu dłuższych skryptów, w których nie widać na pierwszy rzut oka co który fragment robi. Komentarze w skryptach PHP są oznaczane tak jak w języku C - znakami /* i */, np:

```
echo("Kawałek
tekstu")
/* Poprzednia linia wyświetla w 2 liniach napis "kawałek tek-
stu". Nie jest to zbyt odkrywcze, ale w końcu to tylko przykład
zastosowania komentarza ;-) */
```

Jak widać komentarz może mieć na kilka linii.

Wykorzystanie zmiennych

W skryptach PHP wykorzystanie zmiennych jest dużo prostsze niż w wielu innych językach. Nazwy zmiennych zawsze zaczynają się od znaku $. Nie jest konieczne deklarowanie zmiennych, tak jak ma to miejsce w wielu innych językach. Aby stworzyć zmienną wystarczy nadać jej jakąś wartość, np:

```
$a=7;
$b="Jakiś tekst";
$c=2.654;
$d=0.0
```

PHP obsługuje następujące typy zmiennych:
integer - liczba całkowita
double - liczba rzeczywista
string - tekst
array - tablica
object - złożone zmienne definiowane przez użytkownika
pdfdoc (Tylko przy włączonej obsłudze dokumentów PDF)
pdfinfo (Tylko przy włączonej obsłudze dokumentów PDF)
Typ zmiennej jest określany automatycznie na podstawie przypisywanej wartości. I tak w powyższym przykładzie $a ma typ integer, $b ma typ string a $c i $d mają typ double (0 jest co prawda liczbą całkowitą, ale każda liczba zawierająca kropkę jest traktowana jako rzeczywista).

Jak widać na powyższym przykładzie, tekst powinien być zawsze ujęty w cudzysłowy. Jeżeli chcemy w tekście umieścić cudzysłów, należy poprzedzić go znakiem \. To samo dotyczy znaku $. W celu umieszczenia wewnątrz tekstu znaku \ należy napisać \\ (znak ucieczki). Aby umieścić w tekście znaku nowej linii można użyć sekwencji \n. Wewnątrz tekstu można też użyć zdefiniowanych wcześniej zmiennych:

```
$a=3;
$b="Jakaś wartość";
$c="$a, $b";
```

Zmienna $c będzie miała wartość "3, Jakaś wartość".

Tablice

Tablica to wiele zmiennych ułożonych kolejno, do których można dostać się za pomocą indeksu. Tak samo jak w przypadku zwykłych zmiennych, aby stworzyć tablicę wystarczy przypisać wartość któremuś z jej pól:

```
$tablica[0]="wartość pola 0";
$tablica[1]="wartość pola 1";
$tablica[2]="wartość pola 2";
$tablica[3]="wartość pola 3";
$tablica[4]="wartość pola 4";
```

Jako indeksu można użyć innej zmiennej, np.

```
$indeks=3;
$tablica[$indeks]=27;
```

Operatory i wyrażenia

Występujące w PHP operatory można podzielić na na kilka grup:

- Arytmetyczne (+, -, /, *, %)
- Bitowe (&, |, ~, ^, >>, <<)
- Logiczne (&&, ||, !)
- Przypisania (=, +=, -=, *=, /=, %=, .=, <<=, >>=, &=, |=, ^=)
- Relacyjne (==, ===, <>, !=, !==, >, <, >=, <=)
- Inkrementacji / dekrementacji (++, --)
- Pozostałe
    - Łańcuchowy – wyrażany za pomocą znaku kropki
    - Warunkowy: (wyra żenie1) ? (wyrażenie2)
    - Kontroli błędów
    - Wykonania polecenia zewnętrznego (` `)
    - Kontroli typów (instanceof)
    - Rzutowania typów – konwersji ((int), (float), (string), (array), (object))
    - Tworzenia obiektów (New nazwa_klasy(argumenty konstruktora))

- Rozdzielania wyrażeń (, -przecinek)

Poprzedni przykład ograniczał się tylko do przypisania kilku zmiennym wartości. Oczywiście można na nich wykonywać działania:

```
$a=5;
$b=$a+2; /* $b ma wartość 7 */
$b=$b+3; /* teraz $b ma wartość 10 */
$b+=3;   /* to ma efekt taki sam jak $b=$b+3, z tym że
            jest wyraźniejsze w zapisiei szybciej działa */
$c=2*$a+3*($b-$a);
```

W jednej lini można umieścić kilka przypisań, np:

```
$a=$b=5;
```

Przypisania są obliczane od prawej strony (w tym wypadku najpierw zmiennej $b przypisywana jest wartość 5, potem zmiennej $a wartość zmiennej $b).

PHP obsługuje kilka typów porównań. Porównanie ma wartość 1 jeżeli warunek jest spełniony, lub 0 gdy nie jest. $a==$b - spełnione gdy $a i $b są równe. Należy pamiętać że $a=$b jest operacją przypisanie - zmiennej $a jest przypisywana wartość $b, i całe wyrażenie ma wartość $b.

$a>$b - spełnione gdy $a jest większe od $b
$a>=$b - spełnione gdy $a jest większe lub równe $b
$a<$b - spełnione gdy $a jest mniejsze od $b
$a<=$b - spełnione gdy $a jest mniejsze lub równe $b

Przykład użycia operatora wykonania polecenia zewnętrznego

```
<?php
$pliki = `ls –la`;
//w wersji dla Windows: $pliki = `dir`;
echo (”<PRE>”);
echo($pliki);
echo(”</PRE>”);
?>
```

Tworzenie funkcji

W programach przykładowych były już użyte funkcje, chociaż nie było o tym mowy. Konkretnie była użyta funkcja echo. Jest to przykład funkcji zdefiniowanej przez twórców języka PHP. Ale można też stworzyć własną funkcję. Wygląda to tak:

```
Function f($a, $b)
{
```

```
      $a+=$b;
      echo($a);
   }

   f(7, 2); /* teraz wykonają się komendy zawarte w treści funkcji.
Zmienna $a będzie miała wartość 7, $b 2 */
```

Jak widać deklaracja funkcji zaczyna się od słowa Function. Następnie podaje się nazwę funkcji i w nawiasie listę parametrów oddzielonych przecinkami. Potem w nawiasach klamrowych należy podać treść funkcji. W celu wywołania funkcji podaje się jej nazwę i w nawiasach listę wartości parametrów. Jeżeli nie przekazuje się żadnych parametrów, i tak należy po nazwie funkcji umieścić pusty nawias.

Funkcji należy używać, gdy dany fragment kodu musi zostać wykonany w wielu miejscach. Dobrze jest też dłuższe fragmenty skryptów umieścić w kilku funkcjach w celu zwiększenia przejrzystości.

Zmienne globalne

Zmienne te pozwalają na uzyskanie najróżniejszych informacji konfiguracyjnych. Poniżej lista dostępnych zmiennych globalnych:

- $GLOBALS
- $_SERVER
- $_GET
- $_POST
- $_COOKIE
- $_FILES
- $_ENV
- $_REQUEST
- $_SESSION

Zmienne w funkcjach

Kiedy zmienna jest zadeklarowana poza funkcją, jej wartość nie będzie widoczna w funkcji. Ilustruje to przykład:

```
   $a=5;
   Function f()
   {
      echo($a);
   }
   f();
```

Wykonanie powyższego kodu nie spowoduje, jak mogłoby się wydawać, wyświetlenia liczby 5. Żeby zmienna globalna (w tym wypadku $a) była widoczna wewnątrz funkcji, należy użyć polecenia global:

```
   $a=5;
   Function f()
   {
```

```
    global $a;
    echo($a);
}
f();
```

Po wykonaniu powyższego kodu w dokumencie pojawi się liczba 5.

Jeżeli stworzymy zmienną wewnątrz funkcji, jej wartość będzie za każdym wywołaniem ustawiana od początku:

```
Function f()
{
    $a=2;
    echo($a);
    $a++;
}
f();
f();
```

Wykonanie tego kodu spowoduje pojawienie się dwa razy tego samego. Jeżeli chcemy, żeby zmienna nie traciła wartości po zakończeniu funkcji, należy użyć polecenia static:

```
Function f()
{
    static $a=2;
    echo($a);
    $a++;
}
f();
f();
```

Teraz w dokumencie pojawi się najpierw liczba 2, potem 3.

Zwracanie wartości

Funkcja może zwrócić wartość. używa się do tego polecenia return:

```
Function f($p)
{
    return 3*$p;
}
$a=f(7);
echo($a); /* $a ma wartość 21 */
/* można też od razu echo(f(7)); */
```

Ten skrypt wyświetli liczbę 21. Po zwróceniu wartości kończy się wykonanie funkcji:

```
Function f()
{
   return 3;
   echo("Ten tekst się nie wyświetli");
}
```

Po wykonaniu polecenia return kończy się działanie funkcji i funkcja echo nie jest wywoływana.

Instrukcje include oraz require

Wywołanie instrukcji include: include(”nazwa pliku”);
Wywołanie instrukcji require(”nazwa pliku”);

Przykład zastosowania instrukcji include:

Napiszemy skrypt, którego zadaniem będzie wyświetlenie dowolnej informacji w przeglądarce taki sposób, iż struktura strony HTML będzie w jednym pliku, natomiast instrukcje echo w drugim

Plik skrypt.php

```
<PHP
echo(” <H2>WITAMY NA STRONIE</H2>”);
?>
```

Teraz utworzymy drugi plik, który będzie zawierał instrukcję include

```
<php
Include ”skrypt.php”
?>
```

Instrukcja require ma działanie bardzo podobne do include, wczytuje plik zewnętrzny o wskazanej nazwie. Różnica ujawnia się w momencie, kiedy wybrany plik nie może zostać wczytany. W takiej sytuacji w przypadku include wygenerowane zostanie ostrzeżenie, ale skrypt podstawowy (wywołujący) będzie kontynuował działanie. Tymczasem w przypadku require skrypt wywołujący zakończy działanie zgłaszając błąd.

Zadania

1. Napisz przykładowy skrypt ilustrujący działanie operatorów bitowych
2. Napisz przykładowy skrypt ilustrujący działanie kilku złożonych operatorów przypisania

3. Napisz przykładowy skrypt ilustrujący zasadę działania operatorów (np. 2+3*6)

4. Napisz przykładowy skrypt, który wyświetli wartości zmiennych globalnych (super-globalnych)