PHP7

TABLICE INFORMATYCZNE • Marcin Lis

Helion

UMIESZCZANIE SKRYPTÓW W KODZIE HTML

Skrypty PHP są najczęściej wykonywane przez serwer WWW (choć nie wyklucza to zastosowań lokalnych). W celu wyodrębnienia kodu skryptu od innych elementów (np. kodu HTML) niezbędne jest umieszczenie go wewnątrz znaczników. Przetwarzane będą jedynie te fragmenty kodu, które znajdą się pomiędzy znacznikiem otwierającym i zamykającym. W PHP7 do dyspozycji są dwa typy znaczników:

- znaczniki kanoniczne,
- znaczniki typu SGML (skrócone).

Znaczniki kanoniczne

Znaczniki kanoniczne to standardowe i najczęściej spotykane znaczniki PHP w postaci:

```
<?php
  //kod skryptu
?>
```

Są one zawsze rozpoznawane niezależnie od tego, jakie opcje włączy się w pliku konfiguracyjnym. Zaleca się stosowanie wyłącznie takich znaczników. Jeżeli plik zawiera jedynie kod skryptu, można pominąć znacznik zamykający:

```
<?php
  //kod skryptu
```

Dzięki temu poza trybem PHP nie pozostaną żadne niepotrzebne białe znaki (spacje, entery, tabulatory).

Znaczniki typu SGML

Znaczniki typu SGML mają następującą postać:

```
<?
  //kod skryptu
?>
```

Jest to najkrótsza forma znaczników bloku PHP, jaką można zastosować (wymaga to umieszczenia w pliku konfiguracyjnym *php.ini* linii `short_open_tag = On`). Nie należy ich stosować w przypadku zagnieżdżania kodu PHP w plikach XML i XHTML.

Inne typy znaczników

Wcześniejsze wersje PHP obsługiwały również znaczniki typu ASP (`<%`, `%>`) i znaczniki skryptów HTML (`<script language="php">`, `</script>`), które w PHP7 zostały wycofane i nie należy ich stosować.

Wyprowadzanie danych

W celu wysłania danych do standardowego wyjścia (pot. wysłanie do przeglądarki, wyświetlenie) używa się instrukcji `echo`:

```
echo ("Witam na stronie!");
echo (100);
```

Zamiast instrukcji `echo` można też zastosować `print`:

```
print ("Witam na stronie!");
print (100);
```

Nawias okrągły może być pominięty:

```
echo "Witam na stronie!";
print 100;
```

Różnica jest taka, że `print` zachowuje się jak funkcja, natomiast `echo` jest tylko konstrukcją języka. W związku z tym `print` może być użyte w złożonych wyrażeniach, a `echo` — nie. Każdą instrukcję kończy się średnikiem.

Przykład skryptu

Najprostszy skrypt po prostu wyświetla napis na ekranie:

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

Po zagnieżdżeniu takiego skryptu w kodzie HTML5 uzyskamy:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Moja strona</title>
  </head>
  <body>
  <?php echo "Mój pierwszy
skrypt"; ?>
  </body>
</html>
```

Instrukcja `echo` istnieje też w wersji skróconej. Zapis wygląda tak:

```
<?=wyrażenie?>
```

co jest odpowiednikiem konstrukcji:

```
<?php echo wyrażenie ?>
```

np.

```
<?=1 + 2 * 3?>
<?="Mój pierwszy skrypt"?>
```

Po zagnieżdżeniu we fragmencie kodu HTML mogłoby to wyglądać tak:

```
<div>
  <p><?="Mój pierwszy
skrypt"?></p>
</div>
```

Zapis skrócony dostępny jest standardowo we wszystkich wersjach PHP, począwszy od 5.4.0. W wersjach wcześniejszych konieczne jest włączenie w pliku konfiguracyjnym opcji `short_open_tag`.

KOMENTARZE

W kodzie PHP można zastosować trzy rodzaje komentarzy — dwa zapożyczone ze składni języków takich jak C i C++ oraz jeden stosowany w powłokach uniksowych.

Komentarz blokowy

Ten typ komentarza zaczyna się od sekwencji znaków `/*`, a kończy sekwencją `*/`. Wszystko to, co znajduje się pomiędzy nimi, zostanie zignorowane przez analizator składniowy PHP:

```
<?php
  /*
  To jest komentarz blokowy
  */
?>
```

Komentarz jednowierszowy

Komentarz tego typu rozpoczyna się od znaków `//` i obowiązuje do końca bieżącego wiersza.

```
<?php
  //To jest komentarz jednowierszowy
?>
```

Komentarz jednowierszowy uniksowy

Komentarz tego typu rozpoczyna się od znaku `#` i obowiązuje do końca bieżącego wiersza.

```
<?php
  #To jest komentarz jednowierszowy
?>
```

ELEMENTY JĘZYKA

Typy danych

Występujące w PHP typy danych można podzielić na:

- typy proste (skalarne, ang. *primitive types*, *scalar types*),
- typy złożone (ang. *compound types*, *complex types*),
- typy specjalne (ang. *special types*).

TYPY PROSTE

Typ boolean

Typ logiczny przyjmujący jedną z dwóch wartości: `true` (prawda) lub `false` (fałsz). W przypadku konwersji innych typów na typ `boolean` obowiązuje zasada, że wartość `false` powstaje z przekształcenia:

- typu `integer` o wartości `0`,
- typu `float` o wartości `0.0`,
- typu łańcuchowego o wartości pustej `""`,
- typu łańcuchowego o wartości `"0"`,
- tablicy o zerowej liczbie elementów,
- typu obiektowego o zerowej liczbie elementów,
- typu specjalnego `null`,
- obiektu `SimpleXML`, utworzonego z pustych znaczników.

W każdym innym przypadku w wyniku konwersji otrzymywana jest wartość `true`.

Typ integer

Typ całkowitoliczbowy (oznaczany jako `int`) reprezentuje zarówno dodatnie, jak i ujemne liczby całkowite. Liczby te mogą być zapisane w formatach: dziesiętnym, ósemkowym (oktalnym), szesnastkowym (heksadecymalnym) i, począwszy od PHP 5.4.0, dwójkowym (binarnym). Domyślnie stosowany jest format dziesiętny. Aby zapisać liczbę ósemkową, należy poprzedzić ją znakiem `0` (zero), liczbę szesnastkową należy poprzedzić znakami `0x` lub `0X`, liczbę dwójkową znakami `0b`:

- `16` to liczba w systemie dziesiętnym o wartości 16,
- `020` to liczba ósemkowa o wartości 16 dziesiętnie,
- `0x10` to liczba szesnastkowa o wartości 16 dziesiętnie,
- `0b10000` to liczba dwójkowa o wartości 16 dziesiętnie.

Maksymalny zakres typu całkowitego zależy od platformy sprzętowo-systemowej, na jakiej uruchamiane jest PHP. Typowe zakresy wartości:

System	Liczba bajtów	Min.	Maks.
32-bitowy	4	-2^{31}	$2^{31}-1$
64-bitowy	8	-2^{63}	$2^{63}-1$

Rozmiar typu w danej implementacji może być odczytany ze stałej `PHP_INT_SIZE`, maksymalna dopuszczalna wartość — ze stałej `PHP_INT_MAX`, natomiast minimalna wartość (począwszy od PHP 7.0.0) — ze stałej `PHP_INT_MIN`. W przypadku przekroczenia zakresu wartość jest konwertowana na typ `float`.

Przy konwersji z typów `boolean` i `float` na typ `integer` obowiązują następujące zasady:

- Konwersja z typu `boolean` o wartości `true` daje w wyniku 1.
- Konwersja z typu `boolean` o wartości `false` daje w wyniku 0.
- Konwersja z typu `float` powoduje zaokrąglenie w dół do najbliższej liczby całkowitej.

Uwaga: jeśli wartość typu `float` przekracza zakres liczb typu `integer`, wartość, jaka powstaje w wyniku konwersji, jest nieokreślona.

W przypadku konwersji z typu `string` obowiązują następujące zasady:

- Jeśli ciąg znaków rozpoczyna się od prawidłowej liczby całkowitej nieprzekraczającej dopuszczalnego zakresu (np. `"256"`, `"-512"`, `"64xyz"`), wynikiem jest wartość reprezentowana przez ten ciąg.
- Jeśli ciąg znaków rozpoczyna się od prawidłowej liczby zmiennopozycyjnej (np. `"1.2"`, `"-2.4"`, `"3.6e2"`, `"2.8e4xyz"`) lub przekracza dopuszczalny zakres, najpierw wykonywana jest konwersja do typu `float`.
- Jeśli ciąg znaków nie rozpoczyna się od prawidłowej liczby, wynikiem jest wartość zero.

Konwersja z innych typów niż wymienione nie została zdefiniowana i takiej konwersji nie należy wykonywać.

Typ float

Typ `float` (oznaczany również jako `double`) reprezentuje liczby zmiennopozycyjne (ang. *floating point*). Ich zakres, podobnie jak dla typu `integer`, zależy od platformy sprzętowo-systemowej (z reguły maksymalna wartość jest zbliżona do 1.8e308). Typowy jest zapis z kropką dziesiętną, czyli np. `1.5`. Dostępna jest też notacja wykładnicza w postaci `XeY`, gdzie `X` jest liczbą w notacji z kropką dziesiętną, a `Y` to żądana potęga liczby 10, np.:

- `1.2e1` to 12,
- `4e2` to 400,
- `0.12e4` to 120.

Podczas konwersji do typu zmiennoprzecinkowego ze wszystkich typów (z wyjątkiem łańcuchowego) najpierw wykonywana jest konwersja do typu `integer`, a następnie z `integer` do `float` (przy próbie wykonania konwersji z typu obiektowego zostanie wygenerowane ostrzeżenie). Przy konwersji z typu `string` obowiązują następujące zasady:

- Jeśli ciąg znaków rozpoczyna się od prawidłowej liczby całkowitej (np. `"256"`, `"-512"`, `"64xyz"`), której wartość nie przekracza dopuszczalnego zakresu dla typu `integer`, najpierw jest wykonywana konwersja do typu `integer`.
- Jeśli ciąg znaków rozpoczyna się od prawidłowej liczby zmiennopozycyjnej (np. `"1.2"`, `"-2.4"`, `"3.6e2"`, `"2.8e4xyz"`) bądź też zaczyna się od liczby całkowitej przekraczającej dopuszczalny zakres dla typu `integer`, wynikiem jest wartość reprezentowana przez ciąg.
- Jeśli ciąg znaków nie rozpoczyna się od prawidłowej liczby, wynikiem jest wartość `0`.

Typ string

Typ `string` to typ łańcuchowy, który służy do zapamiętywania sekwencji bajtów (maksymalnie 2 GB). Pojedynczy znak zapamiętywany jest na jednym bajcie, nie ma więc bezpośredniej obsługi standardu Unicode (jest to ciąg bajtów, a nie znaków). Łańcuch znaków można utworzyć w jednym z podanych niżej sposobów zapisu.

Znaki apostrofu

Ciągi znaków ujęte w znaki apostrofu prostego nie są interpretowane przez PHP i są wyświetlane w niezmienionej postaci:

```
'Przykładowy łańcuch znaków'
```

Wyjątkiem jest jedynie sam znak apostrofu. Aby go uzyskać, należy poprzedzić go znakiem lewego ukośnika (backslash):

```
'Znak apostrofu wygląda tak:
\''
```

Znaki cudzysłowu

Ciągi znaków ujęte w cudzysłów prosty są przetwarzane przez PHP, a występujące w nich zmienne zamieniane na wartości tych zmiennych. Obowiązują przy tym następujące zasady:

- Jeżeli zmienna jest typu znakowego, zawarty w niej ciąg znaków jest wklejany do ciągu bieżącego.
- Jeżeli zmienna jest innego typu niż znakowy, najpierw następuje jej konwersja na typ `string`, a następnie zostaje ona wklejona do ciągu bieżącego.
- Jeżeli zmienna nie zawiera żadnej wartości, jest traktowana jak pusty ciąg znaków.

W ciągach tego typu można stosować następujące sekwencje specjalne:

Sekwencja	Znaczenie
`\n`	nowy wiersz
`\r`	powrót na początek wiersza
`\t`	tabulator poziomy
`\v`	tabulator pionowy
`\e`	ucieczka (Escape)
`\f`	przewinięcie strony
`\\`	ukośnik
`\$`	znak dolara
`\"`	znak cudzysłowu

Jeśli ukośnikiem zostanie poprzedzona liczba, zostanie ona potraktowana jako kod znaku w notacji oktalnej (ósemkowej). Przykładowy ciąg:

```
"\145\143\150\157"
```

zostanie potraktowany tak samo jak napis:

```
"echo"
```

Jeśli przed liczbą pojawi się sekwencja `\x`, zostanie ona potraktowana jako kod znaku w notacji heksadecymalnej (szesnastkowej). Przykładowy ciąg:

```
"\x65\x63\x68\x6F"
```

zostanie potraktowany tak samo jak napis:

```
"echo"
```

Jeśli w ciągu pojawi się sekwencja `\u{kod}`, zostanie ona potraktowana jako kod znaku notacji Unicode (kodowanie UTF-8; tylko w PHP 7.0.0 i wyższych). Przykładowy ciąg:

```
"\u{65}\u{63}\u{68}\u{6F}"
```

zostanie potraktowany tak samo jak napis:

```
"echo"
```

Składnia heredoc

W przypadku składni `heredoc` łańcuch znakowy rozpoczyna się od sekwencji `<<<`, po której następuje identyfikator. Następnie wykorzystuje się ten identyfikator, aby zasygnalizować koniec łańcucha znakowego.

```
<<<ID1
Przykładowy ciąg znaków
ID1;
```

Nazwa identyfikatora musi się zaczynać od znaku podkreślenia lub litery oraz może zawierać dowolną kombinację liter, cyfr i znaków podkreślenia.

Linia kończąca nie może zawierać żadnych innych znaków oprócz identyfikatora i średnika. Uwaga ta dotyczy wszystkich znaków, również spacji, tabulatorów itp.

Składnia nowdoc

Składnia typu `nowdoc` została wprowadzona w PHP 5.3.5. Ciągi tego typu nie są przetwarzane przez PHP (jak przy