Kurs WWW

Paweł Rajba

pawel@ii.uni.wroc.pl
http://pawel.ii.uni.wroc.pl/

Spis treści

Wprowadzenie

- PHP – akronim rekursywny
  - “PHP: Hypertext Processor”
- Tworzony na zasadach OpenSource, bezpłatny
- Dostępny
  - jako moduł lub CGI
  - pod wieloma platformami, m.in. Linux i Windows
  - z wieloma serwerami HTTP, m.in. Apache i IIS
- Można zagnieżdżać znaczniki HTML
- PHP może generować dowolny kod (np. JPEG)

Pierwszy przykład

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
 <title>Przykładzik</title>
</head>
<body>
 <?php echo "Oto przykład pierwszy."; ?>
</body>
</html>
```


Sposoby użycia

- Skrypty po stronie serwera
  - serwer WWW
  - parser PHP
  - przeglądarka
- Skrypty z linii poleceń
  - parser PHP
  - zastosowanie do przetwarzania tekstu
  - uruchamiane w cronie lub menedżerze zadań
- Aplikacje po stronie klienta (np. PHP-GTK)

PHP5 i bazy danych

- Adabas D
- Ingres
- Oracle (OCI7 i OCI8)
- dBase
- InterBase
- Ovrimos
- Empress
- FrontBase
- PostgreSQL
- FilePro
- mSQL
- SolidHyperwave
- Direct MS-SQL
- Sybase
- IBM DB2
- MySQL
- Velocis
- Informix
- ODBC
- Unix dbm

Osadzanie skryptów

- PHP poza trybem "PHP" przepisuje tekst bez zmian
- Sposoby ,,włączania" trybu PHP są następujące:
  - `<? echo "To jest skrypt"; ?>`
  - `<?= "To jest ten sam skrypt"; ?>`
  - `<?php echo "W XHTML i XML lepiej tak"; ?>`
  - `<% echo "To znowu jest skrypt"; %>`
  - `<%= "To znowu jest ten sam skrypt"; %>`
  - ```
    <script language="php">
      echo "Można też tak";
    </script>
    ```


    ```

# Komentarze

- Komentarze są
  - `//, #` – jednowierszowe
  - `/* ... */` – blokowe

 

# Zmienne

- Zmienne są typowane dynamicznie
- Nazwę poprzedzamy znakiem $
- Wielkie i małe litery w nazwach są rozróżnialne
- Do pobrania referencji do zmiennej jest znak &
  - Referencja może być pobrana tylko do nazwanego wyrażenia
- Wyświetlanie zmiennych
  - echo $zmienna; echo "$a i $b";

 

# Zmienne

- Przykłady:
  - Prosty przykład

```
<?php
$imie="Jan"; $nazwisko="Nowak"; $wiek="44";
echo $imie." ".$nazwisko.", ".$wiek." lata";
echo "$imie $nazwisko, $wiek lata"
?>
```

  - Drugi przykład

```
<?php $temat="bieg"; echo "za{$temat}anie"; ?>
```

  - Trzeci przykład

```
<?php $a=1; $b=2; $a=&$b; $a=3; echo "$a $b";
// $zle = &(20*30); - tak nie wolno ?>
```


Zmienne

Wybrane funkcje do obsługi zmiennych

- get_defined_vars
- get_resource_type
- doubleval, floatval, strval
- is_array
- is_bool
- is_callable
- is_double
- is_float
- is_int, is_integer
- is_long
- is_numeric
- is_object
- is_real
- is_resource
- is_scalar
- is_string
- is_null
- print_r, settype
- serialize, unserialize
- isset, unset

Stałe

- Definiujemy poprzez funkcję
  - `bool define(string nazwa, mixed wartość [, bool case_insensitive])`
- Sprawdzamy, czy zdefiniowana poprzez funkcję
  - `bool defined (string nazwa_stalej)`
- Pobieramy wartość stałej poprzez funkcję
  - `mixed constant(string name)`
- Tablicę stałych pobieramy poprzez funkcję
  - `array get_defined_constants()`

Stałe

- Przykłady
  - Przykład pierwszy

```
<?php
  define("STALA", "test");
  echo Stala; // Stala i ,,notice''
  define("STALA", "Ahoj!!!", true);
  echo Stala; // test
?>
```

  - Drugi przykład

```
<?php
  define("STALA", "test",true);
  echo(defined('Stala') ? "OK" : "Nie OK");
  print_r(get_defined_constants());
?>
```


Typy danych

- Skalarne
  - boolowski: boolean
  - liczby całkowite: integer
  - liczby zmiennoprzecinkowe: float
  - łańcuchy znaków: string

Typy danych

- Złożone
  - tablice: array
  - obiekty: object
- Specjalne
  - identyfikatory zasobów: resource
  - puste: null

Typy danych

- Boolowski: TRUE,FALSE
- Liczby całkowite: 1234,-123,0123,0xABC
- Liczby zmiennoprzecinkowe: 1.2,3E-2
- Łańcuchy znaków
  - pojedyncze cudzysłowy
  - podwójne cudzysłowy
  - składnia heredoc

Typy danych

- Łańcuchy znaków c.d.
  - Znaki specjalne: `\n, \r, \t, \\, \$, \"`
  - Składnia heredoc

```
$s = <<<EOD
To jest taki\n
trochę dłuższy napis
EOD;
```

  - Offset: `$s{0}, $s[5]`

Tablice

- Konstruktor array
  - array([klucz =>] wartość, ...)
    - klucz – napis lub liczba całkowita
    - wartość – cokolwiek
- Przypisywanie wartości
  - `$tablica[klucz] = wartość;`
  - `$arr[] = wartość;` // kluczem będzie kolejna liczba
- Usuwanie pary klucz-wartość
  - Poprzez funkcję unset($zmienna)

Tablice

- Przykład 1

```
$a = array( 1 => 'jeden',
            2 => 'dwa',
            3=>'trzy' );
unset($a[2]);
/* to tak, jakby utworzyć tablicę:
$a = array( 1 => 'jeden', 3 => 'trzy');
   ale NIE:
$a = array(1 => 'jeden', 2 => 'trzy');
*/
```


Tablice

- Przykład 2:

```
$wypasiona_zmienna = array(
 "owoce" =>
    array("a" => "pomarańcza",
          "b" => "banan",
          "c" => "jabłko"),
 "liczby" => array(1,2,3,4,5,6),
 "dziury" => array(
        "pierwsza",
    5 => "druga",
        "trzecia")
);
$x = $wypasiona_zmienna['dziury'][5];
```


Struktury kontrolne

- Instrukcja warunkowa
  - if, else, elseif
    - `if (cond) instrukcje;`
      `elseif (cond) instrukcje;`
      `else instrukcje;`
    - `(cond) ? expr1 : expr2;`
- Pętle
  - while..do
    - `while (cond) instrukcje;`
  - do..while
    - `do instrukcje while (cond);`

Struktury kontrolne

- Pętle c.d.
  - for
    - `for ( init, cond, post ) instrukcje;`
  - foreach
    - `foreach(tablica as $wartość) wyrażenie`
    - `foreach(tablica as $klucz => $wartość) wyrażenie`
  - break, continue
  - switch
  - return

Struktury kontrolne

- Składnia alternatywna
  - Dostępna dla poleceń if, while, for, foreach i switch
  - Polega na zamianie nawiasu otwierającego { na dwukropek, a nawiasu zamykającego } na słowo endif, endwhile, endfor, endforeach i endswitch odpowiednio.

Funkcje

- Deklaracja
  - `function f($arg1,...,$argN) { kod funkcji; }`
- Instrukcja return zwraca wartość
  - jeśli chcemy, żeby funkcja zwracała referencję:
    ```
    function &f($arg1,...,$argN) { return $ref; }
    $x=&nazwa(1,...,N);
    ```
- Przekazywanie parametrów
  - Przez wartość, czyli normalnie
  - Przez referencję: `nazwa(&$param)`
    - warto, kiedy przekazujemy duże teksty lub tablice

Funkcje

- Parametry domyślne
  - Są dostępne, można przekazywać dowolne typy
  - Przykład

```
function zamowienie($termin, $prod=array("s","d")) {
  return "Zamówienie na: ".join(", ", $prod).". Termin: ".
   ( is_null($termin) ? "nieznany" : $termin )."<br>";
}
echo zamowienie(null);
echo zamowienie("jutro", array("sery"));
```

- Sprawdzanie, czy funkcja jest zdefiniowana
  - **`bool function_exists(string $function_name)`**
    - Przydatne do np. sprawdzenia, czy jest załadowana niezbędna biblioteka

Funkcje

- Zmienne funkcje
  - Jeśli utworzymy zmienną, po czym do zmiennej dodamy nawiasy, PHP będzie szukało funkcji o nazwie wartości zmiennej i będzie chciało ją wywołać
  - Przykład
    - ```
      function func($arg = '')
      {
        echo "Argumentem jest '$arg'.<br />\n";
      }
      $func = 'func';
      $func('test'); // wywoła bar()
      ```
- Dozwolone jest rekurencja


      ```

# Funkcje

- Inne dostępne konstrukcje
  - Funkcje warunkowe

```
if ($warunek) {
  function func() {
    echo "Nie istnieję dopóki wykonanie ";
    echo " programu tutaj nie dojdzie.\n";
  }
}
```

    - funkcja func będzie istnieć tylko wtedy, gdy warunek będzie prawdziwy

Funkcje

- Inne dostępne konstrukcje
  - Funkcje wewnątrz funkcji

```
/* funkcja foo deklaruje funkcję bar */
function foo() {
  function bar() {
    echo "Istnieję od momentu wywołania foo().\n";
  }
}

/* Nie można wywołać bar() ponieważ nie istnieje. */
foo();

/* Teraz można wywołać bar(), ponieważ wywołanie foo()
utworzyło tą funkcję. */

bar();
```


Dołączanie zewnętrznych plików

- Instrukcje
  - include()
  - require()
  - include_once()
  - require_once()
- Czym to się różni?

Obsługa napisów

- Przegląd funkcji
  - void echo string s1 [, string sn ...]
  - string ltrim(string str [, string charlist])
  - string rtrim(string str [, string charlist])
  - string trim(string str [, string charlist])
  - string chr(int ascii)
  - int ord(string str)
  - int strcmp(string str1, string str2)
  - int strpos(string gdzie, string co [, int offset])

Obsługa napisów

- Przegląd funkcji c.d.
  - int strrpos(string gdzie, char co)
  - int strlen(string s)
  - string strtolower(string str)
  - string strtoupper(string str)
  - string strstr(string str1, string str2)
  - string substr(string str, int start [, int length])
  - int substr_count(string str1, string str2)
  - int substr_replace(string s1, string s2, int start [, int length])

Obsługa napisów

- Przegląd funkcji c.d.
  - array explode(string separator, string str [, int limit])
  - string implode(string glue, array pieces)
  - string strrev(string str)
  - string strip_tags(string str [, string dobre_tagi])
  - string wordwrap (string str [, int width
    [, string break [, boolean cut]]])
  - string addslashes (string str)
  - string addcslashes (string str, string charlist)

Obsługa napisów

- Przegląd funkcji c.d.
  - string nl2br(string str)
  - string htmlspecialchars (string string
    [, int quote_style [, string charset]])
    - wymienia znaki: `<, >, &, ", '` na ich kody: `< > & &quot; &#039;`
    - ' wymienia, jeśli włączone ENT_QUOTES
    - " wymienia, jeśli wyłączone ENT_NOQUOTES
  - string md5(string str)
  - string md5_file (string filename [, bool raw_output])
  - string bin2hex (string str)

Obsługa plików

- Przegląd funkcji:
  - int fopen(string nazwa, string tryb)
  - string fgetc(int fp)
  - string fgets(int fp [, int długość])
  - string fgetss(int fp, int długość
    [, string dozwolone_tagi])
    - to samo co fgets, ale dodatkowo usuwa znaczniki HTML
  - string fread(int fp, int długość)
  - mixed fscanf(int fp, string format)
  - array fgetcsv(int fp, int długość [, string delimiter])

Obsługa plików

- Przegląd funkcji c.d.
  - int fwrite(int fp, string napis [, int length])
  - int fputs(int fp, string napis [, int length])
    - to samo co fwrite
  - bool fclose(int fp)
  - int copy(string źródło, string przeznaczenie)
  - bool rename(stara_nazwa, nowa_nazwa)
  - int unlink(string nazwa_pliku)
  - bool file_exists(string nazwa_pliku)
  - int filesize(string nazwa_pliku)

Obsługa plików

- Przegląd funkcji c.d.
  - int ftruncate(int fp, int rozmiar)
  - int mkdir(string pathname, int mode)
  - bool rmdir(string dirname)
  - string dirname(string ścieżka)
  - string basename (string ścieżka [, string przyrostek])
    - po podaniu, przyrostek zostanie odcięty od końca nazwy
  - bool is_dir(string nazwa)
  - bool is_file(string nazwa)

Obsługa plików

- Przegląd funkcji c.d.
  - array pathinfo(string path)
  - int fseek(resource handle, int offset [, int whence])
    - parametr whence można ustawić na
      - SEEK_SET – ustawia kursor na pozycji równej dokładnie offset (domyślne)
      - SEEK_CUR – ustawia kursor na pozycji bieżąca+offset
      - SEEK_END – ustawia kursor na pozycji koniec pliku+offset
    - parametr offset może być ujemny
  - bool rewind(fp), int ftell(fp)
  - string tempnam(string dir, string prefix)

Wyrażenia regularne

- Przegląd funkcji
  - ereg – sprawdza dopasowanie do wyrażenia
    - bool ereg(string pattern, string string [, array &regs])
  - eregi – j.w., tylko funkcja jest case insensitive
  - ereg_replace – wymienia wyrażenie regularne
    - string ereg_replace(string pattern, string replace, string str)
  - eregi_replace – j.w., tylko funkcja jest c. i.
  - split – podobne do explode, działa na wyr. reg.
    - array split(string pattern, string string [, int limit])
  - spliti – j.w., tylko funkcja jest c.i.

Użycie z wiersza poleceń

- Interpreter PHP może być używany z wiersza poleceń
  - używamy wtedy programu php5 lub php.exe
- Sposoby użycia
  - php5 [-f] skrypt.php
    - normalne odpalenie
  - php5 -r 'tekst skryptu'
    - odpalenie parametru -r np. php5 -r 'echo "Hello world!";'
  - php5 -s skrypt.php
    - wygenerowanie HTMLa ładnie formatującego skrypt.php

Użycie z wiersza poleceń

- Użycie opcji -B, -R i -E
  - -B 'kod' – kod odpalany przed przetwarzaniem skryptu ze stdin
  - -R – kod odpalany dla każdego wiersza ze stdin
    - tutaj dostępne są zmienne: argn – aktualny wiersz, argi – numer bieżącego wiersza
  - -E – kod odpalany po przetworzeniu wierszy z stdin
- Przykład (z dokumentacji)
  - ```
    $ find my_proj | php -B '$l=0;'
    -R '$l += count(@file($argn));' -E 'echo "Total Lines: $l\n";'
    Total Lines: 37328
    ```


    ```

# Użycie z wiersza poleceń

- Użycie opcji -F <plik>
  - Opcja służy do uruchomienia skryptu dla każdego wiersza pliku z danymi
  - Tutaj także dostępne są zmienne $argi i $argn
- Przykład
  - Tworzymy plik z danymi dane.txt
  - Skrypt s.php jest treści: echo "$argi: $argn\n"
  - Odpalamy polecenie
    - `cat dane.txt | php5 -F s.php`
    - `php5 -F s.php < dane.txt`

 

# Użycie z wiersza poleceń

- Tworzenie pliku będącego samodzielnym skryptem
  - W pierwszym wierszu wpisujemy `#!/usr/bin/php5`
  - A potem leci normalny skrypt w PHP
- Do skryptu można przekazywać parametry
- Dostęp do parametrów wewnątrz skryptu
  - $argc – liczba parametrów
  - $argv – tablica parametrów
    - nr 0 – nazwa skryptu, nr 1,... – kolejne parametry

 