

PHP – PHP Hypertext Preprocessor

Igor Wojnicki

Katedra Informatyki Stosowanej
Akademia Górniczo-Hutnicza w Krakowie

10 grudnia 2013

\$Id: php1.tex,v 1.1 2011/11/19 19:14:25 wojnicki Exp wojnicki \$

- 1 PHP
 - Optymalizacja
 - Przesyłanie plików
 - Współpraca z Bazami Danych

I. Wojnicki, JiTW

Spis Treści

- 1 PHP
 - Optymalizacja
 - Przesyłanie plików
 - Współpraca z Bazami Danych

I. Wojnicki, JiTW

Debugging

- <http://www.php-debugger.com/dbg/>
Wymaga zainstalowania modułu dla serwera HTTP.
- <http://www.xdebug.org/>
Wymaga odpowiedniej konfiguracji interpretera PHP i instalacji oprogramowania na serwerze.
- FireBug + FirePHP (przesyłanie komunikatów do konsoli FireBug)
- Śledzenie wartości zmiennych:

```
var_dump()  
print_r()
```

Optymalizacja Skryptów PHP I

- FireBug
- Pomiar wydajności w PHP.

```
function get_microtime() {  
    list($usec, $sec) = explode(' ',  
                                microtime());  
    return ((float) $usec + (float) $sec);  
}
```

- `microtime(true)` - **wartość jako liczba rzeczywista.**
0.31258300 1382259058 **vs.** 1382259058.3126
- **Profiler:** <http://www.php-debugger.com/dbg/>

Optymalizacja Skryptów PHP II

- Cytowanie – zamiana podwójnych cudzysłówów na apostrofy daje kilka procent zysku.

```
$zmienna1 = "$a $b $c"; //wolne  
// szybsze o kilka procent:  
$zmienna2 = $a . ' ' . $b . ' ' . $c;
```

Optymalizacja Skryptów PHP III

- Wyświetlanie (generacja standardowego wyjścia)
 - Używaj `echo` – przewaga nad wychodzeniem z trybu PHP znacznikiem `?>` i ponownym wchodzeniem w ten tryb jest niewielka.
 - cudzysłów vs. apostrof – cytowanie zmiennej powoduje znaczne obniżenie wydajności.
 - `printf()` jest znacznie mniej wydajna, bogate możliwości formatowania napisów, używaj tylko wtedy gdy jest to konieczne.

Optymalizacja Skryptów PHP IV

- Niepotrzebne zmienne
- Niepotrzebne funkcje
- Niepotrzebne wywołania funkcji

```
$arr = array(1,2,3,4,5,6,7,8,9,10);  
//count() przy każdym przebiegu!  
for ($i=0; $i<count($arr); $i++) {  
    echo $arr[$i];  
}
```


Optymalizacja Skryptów PHP V

- Programowanie obiektowe (od PHP5.1.0 znacznie lepiej)

I. Wojnicki, JiTW

Optymalizacja Skryptów PHP VI

- Operacje na plikach Wczytanie całego pliku do tablicy:

```
$dane = implode ('', file('plik'));
```

Zabiera dużo pamięci.

```
$plik = file('nazwa');
```

```
$linia = $plik[1];
```

Lepiej zrobić to tak:

```
$fd = fopen('plik', 'r');
```

```
fgets($fd);
```

```
$linia = fgets($fd);
```

```
fclose($fd);
```

Optymalizacja Skryptów PHP VII

- Zapytania SQL – Let Others Do
Pętla w pętli vs. bardziej złożone zapytanie SQL.

I. Wojnicki, JitW

Spis Treści

- 1 PHP
 - Optymalizacja
 - Przesyłanie plików
 - Współpraca z Bazami Danych

I. Wojnicki, JiTW

Przesyłanie plików, XHTML

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Przykładowa obsługa plików</title>
    <meta http-equiv="content-type"
          content="text/html; charset=UTF-8" />
  </head>
  <body>
    <form action="file.php" method="post"
          enctype="multipart/form-data">
      <p>
        <input type="file" name="plik_pierwszy" />
        <input type="file" name="plik_drugi" />
        <input type="submit" value="wyślij" />
      </p>
    </form>
  </body>
```

Obsługa przesyłanych plików, PHP

```
<?php
header('Content-type: text/plain');
var_dump($_FILES);
if (move_uploaded_file(
    $_FILES['plik_pierwszy']['tmp_name'],
    '/tmp/nazwa_pliku_po_przeslaniu'))
    echo 'ok';
else
    echo 'err';
?>
```

`move_uploaded_file()` dodatkowo sprawdza czy plik został przesłany przez HTTP POST.

Rezultat I

```
array(2) {  
  ["plik_pierwszy"]=>  
  array(5) {  
    ["name"]=>  
    string(10) "files.html"  
    ["type"]=>  
    string(9) "text/html"  
    ["tmp_name"]=>  
    string(14) "/tmp/php3sBkza"  
    ["error"]=>  
    int(0)  
    ["size"]=>  
    int(570)  
  }  
}
```

Rezultat II

```
["plik_drugi"]=>
array(5) {
  ["name"]=>
  string(25) "qr-wojnicki-agh-email.png"
  ["type"]=>
  string(9) "image/png"
  ["tmp_name"]=>
  string(14) "/tmp/php4f9qRW"
  ["error"]=>
  int(0)
  ["size"]=>
  int(353)
}
}
ok
```


Inne wartości w \$_FILES

```
$_FILES['nazwa_pola']['name']  
$_FILES['nazwa_pola']['type']  
$_FILES['nazwa_pola']['size']  
$_FILES['nazwa_pola']['tmp_name']  
$_FILES['nazwa_pola']['error']
```

Informacje w \$_FILES przesyłane są zawsze nawet, gdy nie jest przesłany żaden plik (informacja o błędzie: <http://www.php.net/manual/en/features.file-upload.errors.php>).

Pliki na serwerze

- Przy przesyłaniu plików można je przenieść w dowolną lokalizację w systemie plików serwera.
- Właściciel przesłanego pliku, w zależności od konfiguracji:
 - właściciel skryptu,
 - właściciel interpretera.
- Konfigurowalny limit max wielkości przesyłanego pliku!
- Konfigurowalny limit max czasu wykonywania skryptu!

Code/Script Injection

Należy uniemożliwić:

- przesyłanie plików w sposób umożliwiający trafienie do innego katalogu (użycie separatorów nazw katalogów w nazwie pliku)
 - `basename()`
- przesłanie kodu do uruchomienia (np. skryptu PHP)
 - odpowiednia lokalizacja docelowego katalogu, bez możliwości wykonywania kodu (w przypadku PHP bez dostępu dla serwera WWW)

Spis Treści

- 1 PHP
 - Optymalizacja
 - Przesyłanie plików
 - **Współpraca z Bazami Danych**

I. Wojnicki, JiTW

Systemy Baz Danych i Połączenia

- Obsługiwane Systemy Baz Danych:
 - PostgreSQL
 - MySQL
 - ODBC
 - Oracle
 - SQLite
 - mSQL (miniSQL)
 - Microsoft SQL
 - Sybase
 - Informix
- Połączenia Stałe (persistent) vs. Tymczasowe (temporary) – w zależności od RDBMS możliwy zysk wydajności.

Połączenia z bazami danych

- BerkleyDB.
- Natywne (sterowniki/funkcje dla konkretnych baz danych),
- dbx — abstrakcyjny interfejs oparty o funkcje (nie jest częścią podstawowej dystrybucji począwszy od PHP5.1.0).
- PDO — j.w. ale obiektowy.

Połączenie z MySQL I

```
$con_id = mysql_connect('localhost', 'root');
mysql_select_db('Test', $con_id);
$query = 'SELECT * FROM Pracownicy';
$result = mysql_query($query, $con_id);
#wygeneruj otrzymane wyniki w postaci tabeli
echo "<table border=1 align=center>\n";
echo "<tr>\n";
echo "<td>ID</td>\n";
echo "<td>Nazwisko</td>\n";
echo "<td>Imię</td>\n";
echo "<td>Data urodzenia</td>\n";
echo "<td>Stanowisko</td>\n";
#wypisz wszystkie rekordy
while ($myrow = mysql_fetch_row($result)) {
```

Połączenie z MySQL II

```
printf("<tr><td>%s</td><td>%s</td>  
      <td>%s</td><td>%s</td>  
      <td>%s</td></tr>\n",  
      $myrow[1], $myrow[2],  
      $myrow[3], $myrow[4], $myrow[5]);  
}  
mysql_free_result($query);  
mysql_close($con_id);  
echo "</table>\n";
```


Przykład obsługi takiej samej bazy danych ODBC I

```
$con_id = odbc_connect("Test", "tlogin",  
                      "tpassword");  
$query = 'SELECT * FROM Pracownicy';  
$result = odbc_exec($con_id, $query);  
#wygeneruj otrzymane wyniki w postaci tabeli  
echo "<table border=1 align=center>\n";  
echo "<tr>\n";  
echo "<td>ID</td>\n";  
echo "<td>Nazwisko</td>\n";  
echo "<td>Imię</td>\n";  
echo "<td>Data urodzenia</td>\n";  
echo "<td>Stanowisko</td>\n";  
#wypisz wszystkie rekordy  
while (odbc_fetch_row($result)) {
```

Przykład obsługi takiej samej bazy danych ODBC II

```
printf("<tr><td>%s</td><td>%s</td>  
      <td>%s</td><td>%s</td>  
      <td>%s</td></tr>\n",  
      odbc_result($result, 'ID'),  
      odbc_result($result, 2),  
      odbc_result($result, 'Imie'),  
      odbc_result($result, 4),  
      odbc_result($result, 'Stanowisko'));  
}  
odbc_free_result($query);  
odbc_close($con_id);  
echo "</table>\n";
```

Połączenie z PostgreSQL I

```
<?php
$database="verlag";
$db_conn=pg_connect("host=localhost port=5432
    dbname=$database user=me password=blah");
if (!$db_conn): ?>
    <H1>Błąd przy połączeniu z bazą postgres
    <?php echo $database ?></H1> <?php
    exit;
endif;

$qu = pg_query($db_conn,
    "SELECT * FROM verlag ORDER BY autor");
$row = 0;
```

Połączenie z PostgreSQL II

```
while ($data = pg_fetch_object ($qu, $row)) {  
    echo $data->autor." ("  
    echo $data->jahr ."): ";  
    echo $data->titel."<BR>";  
    $row++;  
}  
pg_free_result ($qu);  
pg_close ($db_conn);
```

SQL Injection

- Kuszące jest przesyłanie (części/całości) kodu SQL do wykonania jako wartości atrybutów za pomocą HTTP GET/POST
 - np. fragment dot. sposobu sortowania danych,
 - całość zapytania bezpośrednio budowana na podstawie wartości w formularzu.
- Potencjalnie może to być wykorzystane do uruchamiania dowolnych zapytań na bazie danych.
- Należy *weryfikować wszystkie dane*, które mają być użyte w zapytaniach.