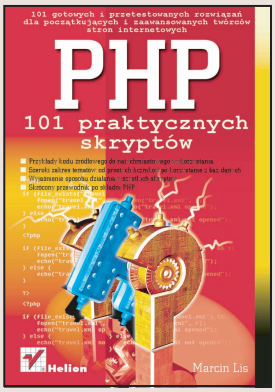
PHP. 101 praktycznych skryptów



Autor: Marcin Lis
ISBN: 83-7361-127-4
Format: B5, stron: około 242

PHP to najpopularniejszy język programowania serwisów internetowych. Łączy on w sobie przyjazną składnię z bogatą biblioteką funkcji, które umożliwiają szybkie tworzenie zarówno prostych skryptów, jak i zaawansowanych, opartych na bazach danych, profesjonalnych aplikacji. Nie bez znaczenia jest też fakt, że jest to jeden z najłatwiejszych do nauczenia się języków programowania, co sprawia, że chętnie sięgają po niego początkujący programiści.

Tytuł tej książki mówi sam za siebie. Znajdziesz w niej 101 skryptów PHP gotowych do użycia w Twoim serwisie internetowym. Początkujący programiści mogą po prostu wklejać prezentowane tu fragmenty kodu na swoje strony, zaawansowani pokuszą się zapewne o ich modyfikację i dostosowanie do własnych potrzeb. Wszystkie skrypty zostały dokładnie objaśnione, można więc zapoznać się z niuansami programowania w PHP.

Przykłady dotyczą:

- Korzystania z funkcji internetowych PHP
- Używania systemu plików do przechowywania danych
- Tworzenia liczników i ksiąg gości
- Łączenia się z bazami danych
- Tworzenia grafiki w PHP
- Funkcji związanych z datą i czasem
- Zabezpieczania stron za pomocą haseł i logowania użytkowników

Cennym uzupełnieniem jest zwięzły opis operatorów, typów danych, instrukcji i wybranych funkcji języka PHP, z którego możesz zawsze skorzystać, gdy czegoś zapomnisz. Książka „PHP. 101 praktycznych skryptów” powinna znaleźć się na półce każdej osoby używającej tego doskonałego narzędzia.

Wydawnictwo Helion
ul. Chopina 6
44-100 Gliwice
tel. (32)230-98-63
e-mail: helion@helion.pl

Spis treści

Rozdział 2.
System plików

Skrypt 18. Wyświetlenie zawartości katalogu

Zawartość danego katalogu możemy wyświetlić, wywołując polecenie danego systemu operacyjnego. Polecenie takie możemy wprowadzić, ujmując je w znaki ` `, np. `ls -la`. Zwróćmy jednak uwagę, że to nie jest zwykły apostrof, tylko tzw. *lewy apostrof*, znajdujący się na klawiaturze pod znakiem tyldy. Wynik wykonania polecenia zewnętrznego można przypisać do zmiennej, a jej zawartość wyświetlić z kolei na ekranie. Tak właśnie postąpimy w niniejszym skrypcie.

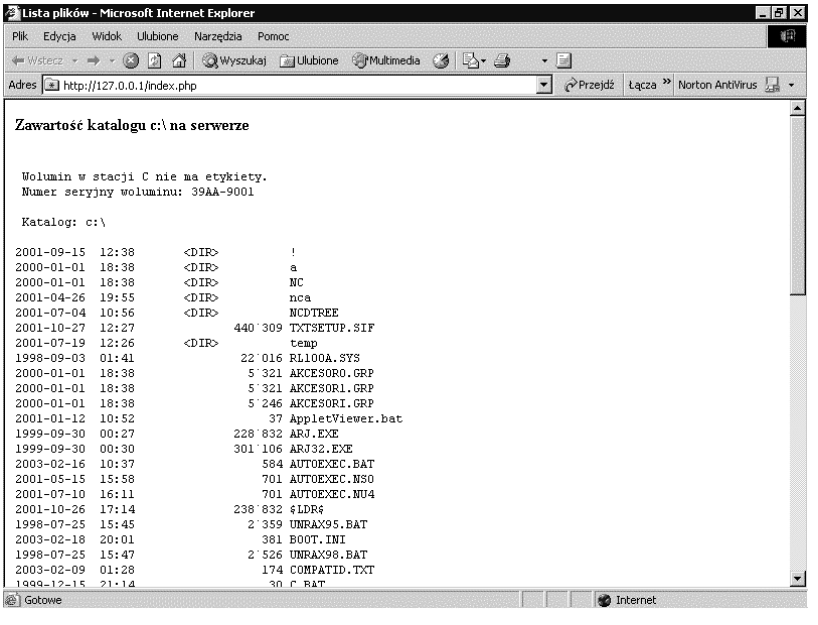
```
<HTML>
<HEAD>
<TITLE>Lista plików</TITLE>
</HEAD>
<BODY>
<?PHP
/*Skrypt pochodzi z książki "PHP. 101 praktycznych skryptów"
  autor: Marcin Lis        e-mail: 101scripts@marcinlis.com*/
function listDir($dirName)
{
  $lista = `dir $dirName`;
  $lista = str_replace("<", "&LT;", $lista);
  $lista = str_replace(">", "&GT;", $lista);
  print("<H3>Zawartość katalogu $dirName na serwerze</H3>");
  print("<BR><PRE>");
  print("$lista");
  print("</PRE>");
}
listDir("c:\\");
?>
</BODY>
</HTML>
```

Za uzyskanie zawartości wskazanego katalogu odpowiada funkcja `listDir`, nazwę tego katalogu przekazujemy jej jako parametr. Zauważmy, że w skrypcie dokonujemy zamiany znaków < oraz > na odpowiednie znaki specjalne interpretowane przez przeglądarkę. Chodzi nam tu o uwzględnianie sytuacji, kiedy skrypt jest wykonywany w systemie Windows i w rezultacie działania polecenia `dir` znajdują się ciągi znaków `<DIR>`.

Przeglądarka interpretuje je wtedy jako znaczniki HTML, co prowadzi do zaburzenia układu strony. Można byłoby w tym przypadku zastosować również konstrukcję:

```
$lista = str_replace("<DIR>", "&LT;DIR&GT;", $lista);
```

Przykładowy wynik działania w przypadku, gdy serwer działa w systemie Windows, widoczny jest na rysunku 2.1.



Rysunek 2.1. *Efekt wykonania polecania dir w skrypcie 18.*

Skrypt 19. Wyświetlenie zawartości katalogu II

Poprzedni skrypt (18.) wyświetlał zawartość wybranego katalogu, miał jednak pewną wadę. Otóż korzystał z polecenia zewnętrznego, a polecenie takie zależne jest od systemu operacyjnego, na którym uruchomiony jest serwer. Przeniesienie skryptu na inną platformę może zatem spowodować konieczność jego modyfikacji, choć np. Linux obsługuje dosowe polecenie `dir`. Co więcej, każdy system może w inny sposób wyświetlać listę.

Jeśli zatem chcemy uzyskać za każdym razem taki sam efekt, należy użyć innego sposobu dostępu do zawartości katalogu. Użyjemy wbudowanych w PHP funkcji operujących na katalogach dyskowych `opendir()` i `readdir()`. Pierwsza z nich dokonuje otwarcia katalogu określonego przez podany w wywołaniu parametr i zwraca identyfikator pozwalający na dalsze operacje. Druga funkcja — `readdir()` — pozwala na odczytanie nazw kolejnych plików.

```
<HTML>
<HEAD></HEAD>
<BODY>
<?PHP
/*Skrypt pochodzi z książki "PHP. 101 praktycznych skryptów"
  autor: Marcin Lis        e-mail: 101scripts@marcinlis.com*/
function getDir($dir)
```

```
{
  $fd = opendir($dir);
  if(!$fd) return false;
  $contents = "<PRE>";
  while (($file = readdir($fd)) !== false){
         $contents .= "$file";
         $contents .= "\n";
  }
  $contents .= "</PRE>";
  closedir($fd);
  return($contents);
}
print(getDir("c://"));
?>
</BODY>
</HTML>
```

Skrypt 20. Nawigacja po katalogach

Czasami nie wystarczy zwykłe wylistowanie zawartości katalogu na ekranie przeglądarki, zaprezentowane w skryptach 18. i 19. Może na przykład zaistnieć potrzeba nawigacji po katalogach. Nic nie stoi na przeszkodzie, aby napisać skrypt udostępniający taką możliwość.

Nazwę katalogu do wyświetlenia będziemy przekazywali do skryptu w postaci parametru o nazwie `dir`, przekazywanego metodą `GET`. Otrzymamy go, odwołując się do indeksu `dir` w globalnej tablicy `$_GET`. W przypadku, gdy parametr nie zostanie podany, zostanie wyświetlona zawartość katalogu bieżącego. Do pobrania zawartości katalogu wykorzystamy znaną z poprzednich skryptów kombinację funkcji `opendir` oraz `readdir`. Tym razem jednak w pętli `while` sprawdzamy, czy aktualnie pobranym elementem jest plik czy katalog. Jeśli jest to plik, wyświetlamy po prostu jego nazwę, jeśli jednak natrafimy na katalog, tworzymy odnośnik HTML.

Odnośnik tworzymy za pomocą znacznika `<A>`. Jego parametr `HREF` określa pełną ścieżkę do aktualnie uzyskanego (przez wywołanie funkcji `readdir`) katalogu. Czyli jeśli na przykład jesteśmy w katalogu o nazwie *d:\a* i natrafiliśmy na podkatalog *obrazy*, odnośnik będzie miał postać:

```
<A HREF="d:\a\obrazy">obrazy</A>
```

Po utworzeniu odnośnika na wszelki wypadek wykonujemy operację usunięcia zdublikowanych znaków (\),za co odpowiada linia `$link = str_replace("\\\\", "\\", $link);`. Duplikaty te mogą pojawiać się przy przekazywaniu parametrów do skryptu, należy się więc przed taką ewentualnością zabezpieczyć.

```
<HTML>
<HEAD></HEAD>
<BODY>
<?PHP
/*Skrypt pochodzi z książki "PHP. 101 praktycznych skryptów"
  autor: Marcin Lis        e-mail: 101scripts@marcinlis.com*/
function listDir($dir)
{
```

```
  chdir($dir);
  $dir = getcwd();
  $fd = opendir($dir);
  if(!$fd) return false;
  while (($file = readdir($fd)) !== false){
        if($file == ".") continue;
        if(is_dir($dir."\\".$file)){
              $link = "<A HREF=\"http://127.0.0.1:80/index.php?";
              $link .= "dir=".$dir."\\".$file."\">".$file."</A>";
              $link = str_replace("\\\\", "\\", $link);
              echo("$link");
        }
        else{
              echo("$file");
        }
        echo("<BR>");
  }
  closedir($fd);
}
@$dir = $_GET['dir'];
if($dir == '') $dir = ".";
listDir($dir);
?>
</BODY>
</HTML>
```

Skrypt 21. Usunięcie zawartości katalogu

Katalog możemy skasować dzięki funkcji `rmdir()`. Niestety, aby operacja taka zakończyła się sukcesem, musi on być pusty. Aby więc skasować katalog wraz z zawartością, musimy napisać własną procedurę. Nazwiemy ją `removeDir()`. Nietrudno się domyślić, że będzie ona działała rekurencyjnie. Co zatem należy zrobić? Przede wszystkim, otworzyć katalog, korzystając z funkcji `opendir()`, a następnie odczytać jego zawartość w pętli `while`. Jeśli natrafimy na plik, po prostu go usuwamy (funkcja `unlink()`), jeśli trafimy na katalog, rekurencyjnie wywołujemy procedurę `removeDir()` z jego nazwą (należy podać pełną ścieżkę dostępu) jako parametrem. Po zakończeniu pętli zamykamy katalog za pomocą funkcji `closedir()` oraz usuwamy go, wywołując funkcję `rmdir()`.

Uwaga: funkcje `unlink()` i `rmdir()` nie zadziałają poprawnie, jeśli katalogi i pliki przekazywane im jako parametry nie będą miały odpowiednio ustawionych praw dostępu!

```
<?PHP
/*Skrypt pochodzi z książki "PHP. 101 praktycznych skryptów"
  autor: Marcin Lis        e-mail: 101scripts@marcinlis.com*/
function removeDir($dir)
{
  @$fd = opendir($dir);
  if(!$fd) return false;
  while (($file = readdir($fd)) !== false){
        if($file == "." || $file == "..") continue;
        if(is_dir($dir."\\".$file)){
              @removeDir($dir."\\".$file);
        }
        else{
              @unlink("$dir\\$file");
        }
```

```
  }
  closedir($fd);
  rmdir($dir);
}
removeDir("nazwa_katalogu");
?>
```

Skrypt 22. Rozmiar katalogu

Bardzo często zachodzi potrzeba ustalenia, ile miejsca na dysku zajmuje zawartość wybranego katalogu. Aby się tego dowiedzieć, należy po prostu zsumować wielkość wszystkich zawartych w nim plików, pamiętając jednak, że katalog może mieć podkatalogi. Należy więc napisać procedurę rekurencyjną.

W rzeczywistości, funkcja sumująca wielkości plików będzie bardzo podobna do funkcji usuwającej pliki i katalogi w skrypcie 21. Należy po prostu zamienić wywołania `unlink` na `filesize`. Za przechowywanie aktualnie zliczonej wielkości odpowiadać będzie zmienna `$size`.

```
<?PHP
/*Skrypt pochodzi z książki "PHP. 101 praktycznych skryptów"
  autor: Marcin Lis        e-mail: 101scripts@marcinlis.com*/
function dirSize($dir)
{
  $size = 0;
  @$fd = opendir($dir);
  if(!$fd) return false;
  while (($file = readdir($fd)) !== false){
         if($file == "." || $file == "..") continue;
         if(is_dir($dir."\\".$file)){
                $size += @dirSize($dir."\\".$file);
         }
         else{
                $size += @filesize("$dir\\$file");
        }
  }
  closedir($fd);
  return $size;
}
echo(dirSize("nazwa_katalogu"));
?>
```

Skrypt 23. Kopiowanie katalogu

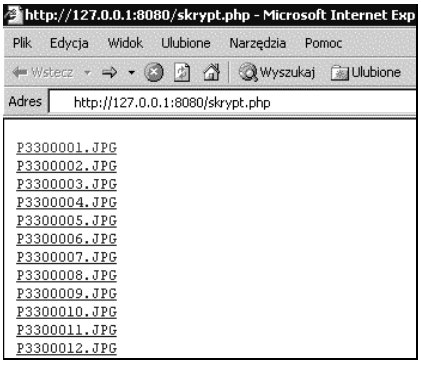
Kopiowanie zawartości jednego katalogu do drugiego to kolejny przykład wykorzystujący funkcje operujące na plikach i katalogach oraz rekurencję. Struktura skryptu jest bardzo podobna do dwóch poprzednich (21. i 22.). Wykorzystujemy jednak kilka dodatkowych funkcji: `file_exists()`, `mkdir()` oraz `copy()`. Pierwsza z nich zwraca wartość `true`, jeśli plik (lub katalog) podany jako parametr istnieje, a wartość `false` w przeciwnym przypadku Druga (`mkdir()`) tworzy katalog o podanej nazwie, natomiast trzecia (`copy()`) wykonuje kopiowanie pliku. Należy zwrócić uwagę, że funkcja `mkdir()` potrafi jedynie utworzyć podkatalog w już istniejącym katalogu. Na przykład instrukcja `mkdir("/pub/pliki/grafika");` oznacza utworzenie podkatalogu o nazwie *grafika* w katalogu */pub/pliki*, który musi istnieć na dysku.

```
<?PHP
/*Skrypt pochodzi z książki "PHP. 101 praktycznych skryptów"
  autor: Marcin Lis        e-mail: 101scripts@marcinlis.com*/
function copyDir($source, $destination)
{
  if(!file_exists($destination)) return false;
  $fd = opendir($source);
  if(!$fd) return false;
  while (($file = readdir($fd)) !== false){
         if($file == "." || $file == "..") continue;
         if(is_dir($source."\\".$file)){
                mkdir($destination."\\".$file);
                copyDir($source."\\".$file, $destination."\\".$file);
         }
         else{
                copy("$source\\$file", "$destination\\$file");
         }
  }
  closedir($fd);
  return true;
}
copyDir("katalog źródłowy", "katalog docelowy");
?>
```

Skrypt 24. Wyświetlenie plików określonego typu jako odnośników

Skrypt 18. pozwalał na wyświetlenie zawartości wybranego katalogu, natomiast skrypt 20. — realizację nawigacji po katalogach. Bardzo przydatną funkcją będzie również wyświetlenie plików o zadanych rozszerzeniach w postaci odnośników (rysunek 2.2).

Rysunek 2.2. *Pliki typu jpg wyświetlone w postaci odnośników*



Zadanie to realizuje funkcja `getDir()`. Jej parametry to nazwa katalogu — `$dir` oraz lista rozszerzeń oddzielonych od siebie znakiem średnika — `$extList`. Za pomocą funkcji `explode()` wyodrębniamy poszczególne rozszerzenia z listy `$extList` i zapisujemy je w tablicy `$arr`. Zawartość wybranego katalogu odczytujemy w pętli `while`, korzystając z funkcji `readDir()`. Z nazwy każdego pliku wyodrębniamy jego rozszerzenie i sprawdzamy, czy odpowiada ono jednej z wartości zapisanych w tablicy `$arr`. Jeśli tak, uwzględniamy dany plik na budowanej liście. Jeśli nie, przechodzimy do kolejnej iteracji pętli. Odnośniki konstruujemy za pomocą typowej konstrukcji ze znacznikiem `<A>`.

```
<?PHP
/*Skrypt pochodzi z książki "PHP. 101 praktycznych skryptów"
  autor: Marcin Lis        e-mail: 101scripts@marcinlis.com*/
function getDir($dir, $extList)
{
  if(!($fd = opendir($dir))){
         return false;
  }
  $arr = explode(";", $extList);
  foreach($arr as $key => $file){
         $arr[$key] = trim($file);
  }
  $contents = "<PRE>";
  while (($file = readdir($fd)) !== false){
         if(($file == "..") || ($file == ".")){
                 continue;
         }
         $ext = substr($file, strlen($file) - 3, 3);
         $ext = strtolower($ext);

         if(in_array($ext, $arr)){
                $contents .= "<A HREF=\"$dir$file\">$file</A>";
                $contents .= "\n";
         }
  }
  $contents .= "</PRE>";
  closedir($fd);
  return($contents);
}
?>
<HTML>
<HEAD></HEAD>
<BODY>
<?PHP print(getDir("./images/", "jpg;png"));?>
</BODY>
</HTML>
```