Współpraca z systemem plików

PHP pozwala na wykonywanie różnych operacji na strukturze systemu plików. Bez problemów można odczytywać rozmaite informacje, np. zawartość wskazanego katalogu lub wielkość pliku, jak również dokonywać modyfikacji samej struktury, czyli tworzyć, usuwać lub przenosić w inne miejsce na dysku pliki i katalogi. Każda taka operacja jest wykonywana przez dedykowaną funkcję.

Odczyt zawartości katalogu

W celu odczytania zawartości wybranego katalogu należy wykonać w sumie trzy operacje:

- Otwarcie katalogu
- Odczyt danych
- Zamknięcie katalogu

Otwarcie katalogu to rezerwacja zasobów oraz informacja dla systemu, że będziemy wykonywać dalsze operacje. Aby otworzyć katalog, używa się funkcji ***opendir***, której wywołanie ma postać:

opendir('nazwa_katalogu')

Funkcja zwraca deskryptor katalogu, czyli specjalny identyfikator, którym należy się posługiwać przy wykonywaniu dalszych operacji.

Odczyt zawartości katalogu, przeprowadza się za pomocą funkcji ***readdir***, której należy przekazać w formie argumentu deskryptor uzyskany przy wywołaniu ***opendir***. Funkcja ***readdir*** działa w taki sposób, że każde jej wywołanie powoduje zwrócenie nazwy kolejnego elementu znajdującego się w katalogu; po odczytaniu wszystkich elementów zwraca ona wartość ***false***.

Gdy odczytana zostanie zawartość katalogu, należy go zamknąć za pomocą funkcji ***closedir***, przekazując jej w postaci argumentu deskryptor otrzymany w wywołaniu ***opendir***.

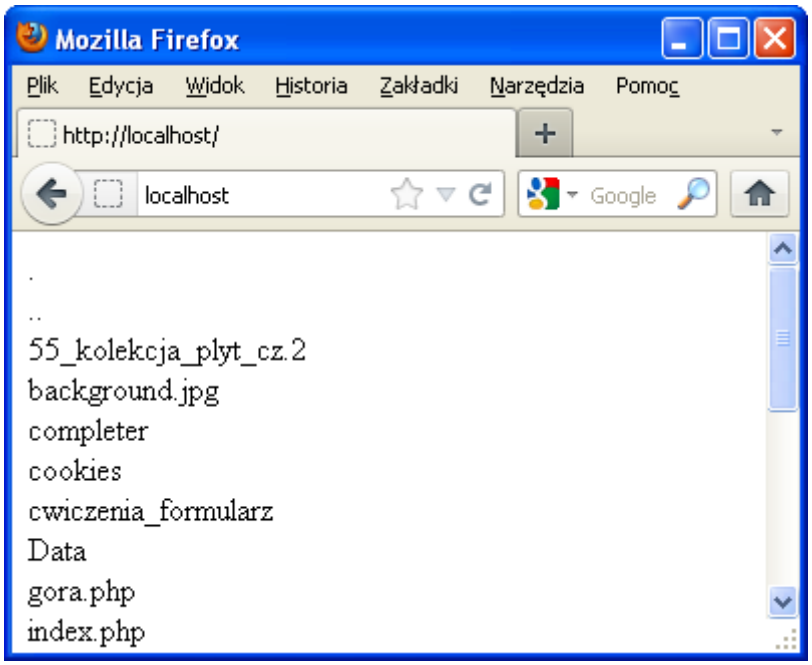
Wykorzystując powyższe informacje, można już w prosty sposób napisać skrypt, który wyświetli zawartość wybranego katalogu.

```php
<?php
$dir = "./";
if(!($fd = opendir($dir)))
  exit("Nie mogę otworzyć katalogu $dir!");

while (($file = readdir($fd)) !== false)
  echo "$file<br \>\n";

closedir($fd);
?>

```



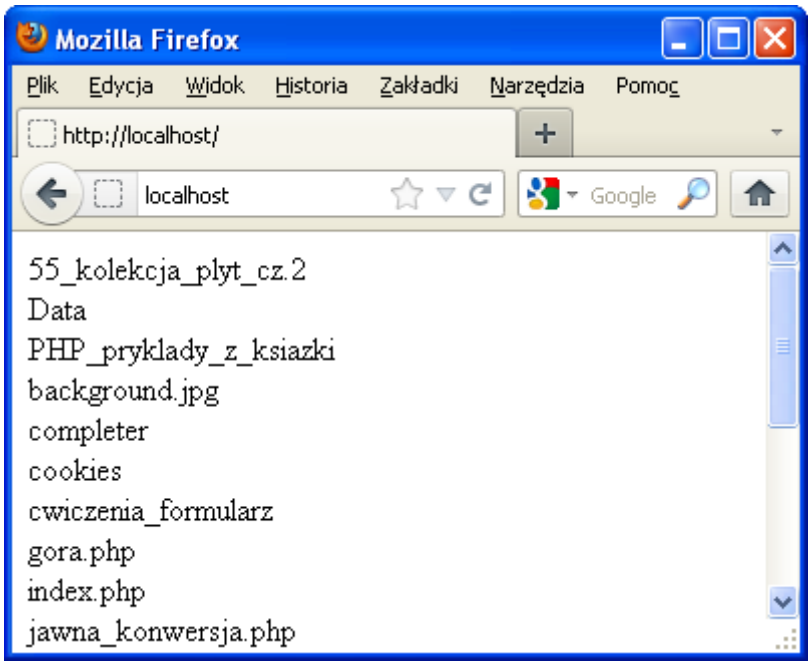
W PHP5 istnieje również funkcja o nazwie ***scandir***, która za jednym wywołaniem pobiera zawartość całego katalogu i zwraca ją w postaci tablicy. Schematyczne wywołanie ma postać:

scandir('nazwa katalogu'[, sortowanie])

Parametr ***nazwa_katalogu*** określa katalog, z którego będą pobierane dane, natomiast sortowanie – sposób sortowania. Ustawienie drugiego parametru na 0 powoduje, że nazwy plików i katalogów będą sortowane alfabetycznie w porządku rosnącym, a ustawienie na 1 lub inną wartość niezerową – że nazwy będą sortowane w porządku malejącym.

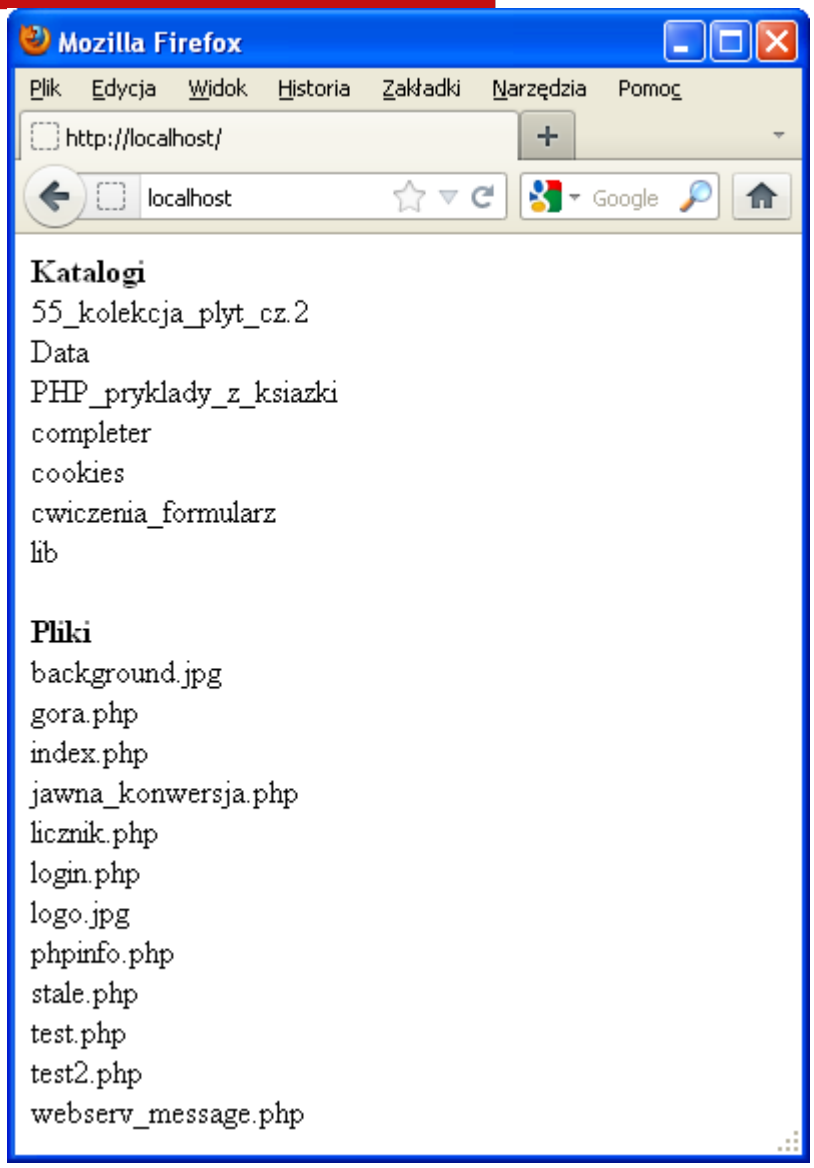
```php
<?php
$dir = "./";
$arr = scandir("$dir");
foreach($arr as $file){
  if($file != '.' && $file != '..'){
    echo "$file <br />\n";
  }
}
?>

```



Ćwiczenie

Wyświetlanie oddzielnych list dla plików i katalogów



Operacje na katalogach

Tworzenie

Do tworzenia katalogów służy funkcja ***mkdir***

mkdir('nazwa'[,tryb[, zagnieżdżone]])

Parametr nazwa określa tu nazwę katalogu, natomiast tryb – prawa dostępu. Argument nazwa może określać zarówno względną, jak i bezwzględną ścieżkę dostępu. Jeśli nie ma określonej ścieżki, nowy katalog zostanie utworzony w katalogu bieżącym. Trzeci parametr ustawiony na ***true*** umożliwia utworzenie zagnieżdżonej struktury katalogów.

Przykładowo, aby założyć podkatalog ***images*** w katalogu ***var/www/,*** należy wykonać instrukcję:

mkdir(„/var/www/images);

Przy czym ta instrukcja zostanie wykonana prawidłowo, jeśli istnieje katalog ***/var/www***. Gdyby go nie było, należałoby skorzystać z trzeciego argumentu, pamiętając, że wtedy nie można pominąć drugiego. Instrukcja miałaby postać:

mkdir(„/var/www/images”,0777,true);

Po jej użyciu zostaną utworzone wszystkie brakujące składowe ścieżki dostępu.

Usuwanie

Do usuwania katalogów służy funkcja ***rmdir***.

rmdir(‘nazwa’)

Usuwany katalog musi być pusty

Zmiana katalogu bieżącego

Do zmiany katalogu bieżącego wykorzystywana jest funkcja ***chdir***, której w postaci argumentu należy przekazać nazwę nowego katalogu bieżącego. Wywołanie ma postać:

chdir(‘nazwa’)

Ustalenie katalogu bieżącego

Jeśli chcemy się dowiedzieć, jaki jest aktualny katalog bieżący, powinniśmy użyć funkcji ***getcwd***, która zwraca jego nazwę w postaci ciągu znaków. Wyświetlenie aktualnego katalogu bieżącego uzyskamy zatem za pomocą instrukcji:

echo getcwd();

Ustalenie, czy katalog istnieje

Aby ustalić czy dany katalog istnieje należy użyć funkcji ***file_exists***, której wywołanie ma postać:

file_exists(‘nazwakatalogu’)

Zwracaną wartością jest ***true***, jeśli katalog istnieje, lub ***false*** w przeciwnym razie. Ta sama funkcja jest wykorzystywana do sprawdzania plików.

Operacje na plikach

Usuwanie plików

Do usunięcia plików służy funkcja ***unlink***, która w postaci argumentu przyjmuje nazwę usuwanego pliku, a zatem jej wywołanie ma schematyczną postać:

unlink(‘./image1.jpg’);

Określenie rozmiaru pliku

Rozmiar pliku można sprawdzić, wywołując funkcje o nazwie ***filesize***, schematycznie:

filesize('nazwapliku')

Funkcja zwraca wartość typu ***integer*** określającą wielkość pliku w bajtach.

Informacje o pliku

Oprócz informacji o wielkości pliku przydatnych może być również kilka innych danych pobieranych za pomocą odpowiednich funkcji. Oto niektóre z nich:

- ***fileatime*** – zwraca znacznik czasu Uniksa określający czas ostatniego dostępu do pliku
- ***filectime*** - zwraca znacznik czasu Uniksa określający czas ostatniej zmiany metadanych pliku.
- ***Filetype*** – zwraca ciąg znaków określający typ pliku. Zwracane wartości to: fifo, char, dir, block, link, file i unknown.
- ***Fileperms*** – zwraca wartość określającą prawa dostępu do pliku
- ***Fileowner*** – zwraca identyfikator właściciela pliku.

Miejsce na dysku

Jeśli chcemy uzyskać informacje o ilości wolnego miejsca na dysku, możemy skorzystać z funkcji ***disk_free_space.*** Przyjmuje ona w postaci argumentu nazwę katalogu i zwraca ilość wolnego miejsca (w bajtach) na dysku logicznym, na którym znajduje się ten katalog. Poprawne są zatem wywołania:

disk_free_space('/')

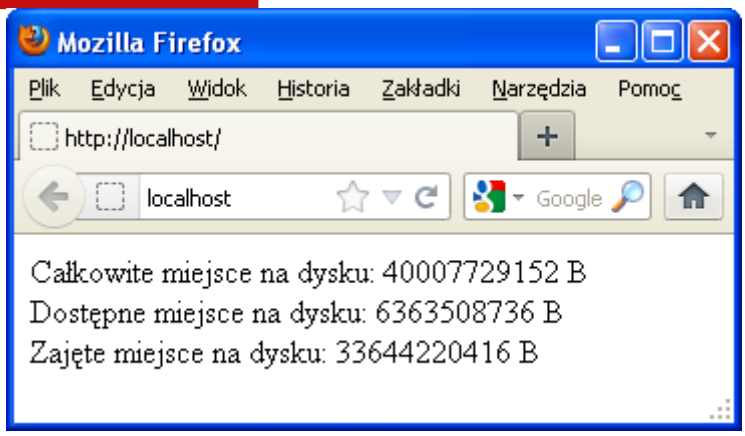
disk_free_space('./')

disk_free_space('/user/tmp')

Jeśli interesuje nas nie wolna, ale całkowita ilość miejsca, zamiast ***disc_free_space*** należy zastosować funkcję ***disk_total_space.***

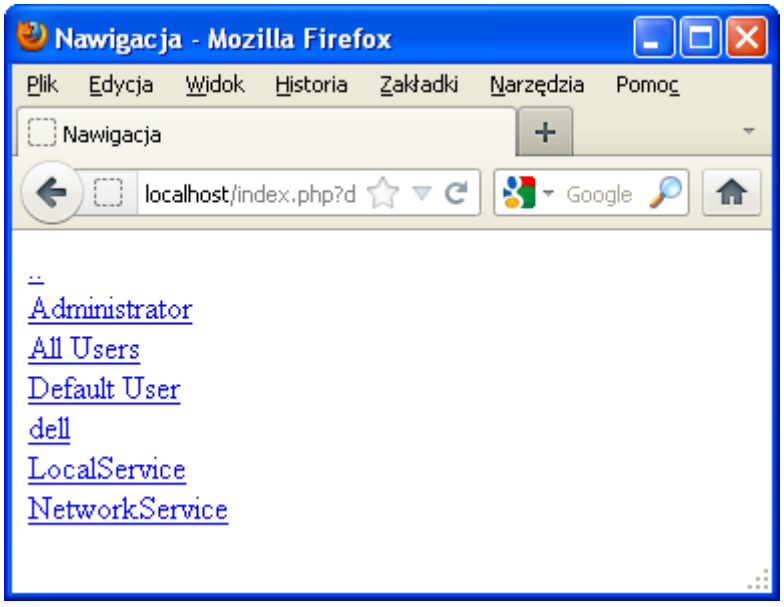
```php
<?php
$wolne_miejsce = disk_free_space("/");
$calkowite_miejsce = disk_total_space("/");
$zajete_miejsce = $calkowite_miejsce - $wolne_miejsce;

echo "Całkowite miejsce na dysku: $calkowite_miejsce B<br />";
echo "Dostępne miejsce na dysku: $wolne_miejsce B<br />";
echo "Zajęte miejsce na dysku: $zajete_miejsce B<br />";
?>
```



Nawigacja po katalogach

Wykorzystując wiadomości przedstawione do tej pory, można w ramach treningu napisać ciekawy skrypt, który będzie pozwalał na przeglądanie w przeglądarce zawartości dysku serwera. Na stronie wyświetlane będą nazwy plików i katalogów. Katalogi będą miały postać odnośników. Kliknięcie takiego katalogu – odnośnika spowoduje wyświetlenie jego zawartości.



```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-2">
<title>Nawigacja</title>
</head>
<body>
<div>
<?php
function listDir($dir)
{
  if(!chdir($dir)){
    echo "Brak uprawnień...";
    return;
  }
  $dir = getcwd();
  if(!$fd = opendir($dir)){
    echo "Brak uprawnień...";
    return;
  }
  $dir = str_replace("\\", "/", $dir);
  if(($count = strlen($dir)) > 0){
    if($dir[$count - 1] == '/'){
      $dir = substr($dir, 0, $count - 1);
    }
  }
  while (($file = readdir($fd)) !== false){
    if($file == ".") continue;
    if(is_dir($dir."/".$file)){
      $path = urlencode($dir."/".$file);
      $link = "<a href=\"http://localhost/index.php?";
      $link .= "dir=$path\">$file</a>\n";
      echo $link;
    }
    else{
     echo $file;
    }
    echo "<br />";
  }
  closedir($fd);
}
if(isSet($_GET['dir'])){
  $dir = $_GET['dir'];
}
else{
  $dir = "/";
}
if($dir == '') $dir = "/";
listDir($dir);
?>
</div>
</body>
</html>
```