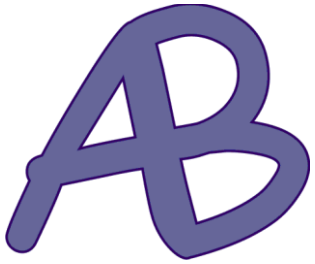


III. PHP

- funkcje,
- pliki

dr Artur Bartoszewski
UTH Radom

Funkcje

Funkcje

Funkcja, która nie zwraca wartości:

```php
<?php
  function nazwa($parametr01, $parametr02) {
  }
?>
```

Funkcja która zwraca wartość”

```php
<?php
  function nazwa_02($parametr01, $parametr02)
  {
      return 100;
  }
?>
```

Domyślne wartości parametrów:

```php
<?php
  function nazwa($parametr01=0, $parametr02=100) {
  }
?>
```

Funkcje

Zmienne statyczne, to zmienne dla których przypisanie wartości zostanie zrealizowane tylko przy pierwszym wywołaniu funkcji, natomiast w kolejnym zapamiętana zostanie ostatnia wartość tej zmiennej

```php
<?php
  function nazwa($parametr01, $parametr02) {
      static $x = 100;
  }
?>
```

Przykład:

```php
function funkcja()
{
    static $x = 0;              // prawidłowo
    static $x = 1 + 2;          // błąd  (w rzeczywistości to jest wyrażenie)
    static $x = sqrt(121);      // błąd  (to również jest wyrażenie)
    $x++;
    echo $x;
}
```

Funkcje – zasięg widoczności zmiennych

Zmienne lokalne, czyli zmienne zadeklarowane wewnątrz funkcji, są widoczne tylko w jej obszarze (co raczej nikogo nie dziwi)

```php
<?php
  function nazwa($parametr01, $parametr02) {
      $x = 100;
  }
?>
```

Zmienne globalne, Zadeklarowane poza funkcją są widoczne poza funkcją lecz nie wewnątrz niej (co już może być pewnym zaskoczeniem dla programistów c++)

```php
$x_globalny = 100;
function nazwa($parametr01, $parametr02)
{
    $x = 100;
    echo $x;
    echo $x_globalny; //błąd - zmienna nie jest widoczna
}
```

Funkcje – zasięg widoczności zmiennych

Są dwa sposoby Na sięgnięcie do zmiennej globalnej z wnętrza funkcji:

- użycie słowa kluczowego ***global***,
- użycie zmiennej super-globalnej ***$GLOBALS['nazwa zmiennej']***

```php
$a = 1;
$b = 2;

function Suma()
{
    global $a, $b;
    $b = $a + $b;
}
```

```php
$a = 1;
$b = 2;

function Suma2()
{
    $GLOBALS['b'] = $GLOBALS['a'] + $GLOBALS['b'];
}
```

Funkcje – zasięg widoczności zmiennych

Tablica $GLOBALS jest asocjacyjną tablicą, w której nazwa zmiennej jest kluczem, a zawartość zmiennej wartością komórki tablicy.
Zauważmy, że $GLOBALS jest dostępna z każdego miejsca, ponieważ $GLOBALS jest tablicą superglobalną.

```php
<?php
function test()
{
    global $HTTP_POST_VARS;
    echo $HTTP_POST_VARS['name'];
    echo $_POST['name'];
}
?>
```

Większość predefiniowanych zmiennych globalna ale nie "superglobalna" i wymaga 'global', by być dostępnymi w zasięgu lokalnym funkcji

Zmienne **Superglobale** są dostępne z każdego miejsca i nie wymagają 'global'.
Superglobale udostępniono wraz z PHP 4.1.0, a HTTP_POST_VARS jest uważane za przestarzałe.

Pliki

Pliki

Otwieranie pliku:

```
$zmienna_plikowa = fopen("nazwa", "tryb");
```

- musimy pamiętać o ścieżce dostępu do naszego pliku,
- jeżeli tryb otwierania jest ustawiony na ”r” czyli tylko do odczytu, to za nazwę pliku może służyć adres URL do pliku.
- należy uważać, by nie stworzyć pliku o nazwie, która już istnieje – istniejący plik zostanie wtedy trwale usunięty w miejsce nowo stworzonego.

Zamykanie pliku (nie jest konieczne):

```
fclose($zmienna_plikowa);
```

Sprawdzenie czy plik istnieje:

```
if (file_exists("nazwa"))
{}
else
{}
```

Pliki – tryby otwarcia

”r”	Otwiera tylko do odczytu; umieszcza wskaźnik pliku na jego początku.
”r+”	Otwiera do odczytu i zapisu; umieszcza wskaźnik pliku na jego początku.
”w”	Otwiera tylko do zapisu; umieszcza wskaźnik pliku na jego początku i obcina plik do zerowej długości. Jeśli plik nie istnieje to próbuje go utworzyć.
”w+”	Otwiera do odczytu i zapisu; umieszcza wskaźnik pliku na jego początku i obcina plik do zerowej długości. Jeśli plik nie istnieje to próbuje go utworzyć.
”a”	Otwiera tylko do zapisu; umieszcza wskaźnik pliku na jego końcu. Jeśli plik nie istnieje to próbuje go utworzyć.
”a+”	Otwiera do odczytu i zapisu; umieszcza wskaźnik pliku na jego końcu. Jeśli plik nie istnieje to próbuje go utworzyć.
”x”	Tworzy i otwiera plik tylko do zapisu; umieszcza wskaźnik pliku na jego początku. Jeśli plik juz istnieje, wywołanie fopen() nie powiedzie się, zwróci FALSE i wygeneruje błąd na poziomie E_WARNING. Jeśli plik nie istnieje, spróbuje go utworzyć. To jest równoważne z określeniem flag O_EXCL|O_CREAT stosowanym w wywołaniu systemowym open.
”x+”	Tworzy i otwiera plik do odczytu i zapisu; umieszcza wskaźnik pliku na jego początku. Jeśli plik już istnieje, wywołanie fopen() nie powiedzie się, zwróci FALSE i wygeneruje błąd na poziomie E_WARNING. Jeśli plik nie istnieje, spróbuje go utworzyć. To jest równoważne z określeniem flag O_EXCL|O_CREAT stosowanym w wywołaniu systemowym open.

Pliki

Wskaźnik pliku określa, skąd w pliku mają być odczytywane dane lub gdzie mają być zapisywane. Przesuwa się on automatycznie dalej po każdej procedurze odczytania określonej porcji danych z pliku.

fseek() - pobiera trzy argumenty:

- uchwyt do pliku,
- przesunięcie,
- (opcjonalny) rodzaj przesunięcia.

Dostępne są trzy wartości przesunięcia:
SEEK_SET – ustawia wskaźnik na pozycję określoną poprzez przesunięcie (domyślne),
SEEK_CUR – ustawia wskaźnik na pozycję równą aktualnej i uwzględnia przesunięcie,
SEEK_END – ustawia wskaźnik na koniec pliku i uwzględnia przesunięcie. Jeśli chcemy ustawić wskaźnik w konkretnej odległości (liczby znaków) od końca zbioru, przesunięcie musi być ujemne.

rewind(), pobiera jako argument uchwyt do pliku i przesuwa wskaźnik na jego początek.

Pliki – odczyt z pliku

Odczyt po kolei po jednym znaku - fgetc()

Zwraca jeden odczytany znak lub false, jeśli natrafimy na koniec zbioru.

```
while (false !== ($char = fgetc($file))) {
    echo "Aktualny znak to $char <br />";
}
```

Ważne jest to, że aby sprawdzić, czy funkcja oddała wartość false, należy użyć operatora === lub !==, ponieważ przy użyciu == lub != wartości „„, (pusty string), lub 0 dzięki zamianie typów zmiennych w PHP też będą interpretowane jako false.

Pliki – odczyt z pliku

Odczyt linijka po linijce fgets().
Jako parametry należy podać uchwyt do pliku i opcjonalnie maksymalną długość odczytanej linii.

```
$plik = fopen("plik.txt","r");
while(!feof($plik))
{
    $wiersz = fgets($plik);
}
```

Pliki – odczyt z pliku

Odczyt całego pliku (za jednym razem) - fread()
argumentami są:

- uchwyt do pliku
- liczba znaków, jaka ma być odczytana.

Często jako liczbę znaków podaje się całkowitą długość pliku, obliczoną funkcją filesize(nazwa_pliku).

```
$plik = fopen("plik.txt", "r");
$dane = fread($plik, filesize("plik.txt"));
```

Lub krócej:

```
$dane = fread(fopen("plik.txt", "r"), filesize("plik.txt"));
```

file_get_contents() - funkcja zwróci całą zawartość pliku |(podobnie jak poprzednia)
jako argument podajemy nazwę pliku – tym razem nie jest to uchwyt, czyli
pliku nie trzeba otwierać, używając fopen()

```
$dane = file_get_contents("plik.txt");
```

Pliki – odczyt z pliku

Przeniesienie zawartości pliku do tablicy, gdzie każdy wpis odpowiada jednej linii z pliku - funkcja file()
W parametrze wystarczy podać nazwę pliku.
Plik nie musi być otwarty funkcją fopen().

```
$tablica = file("plik.txt");
    foreach($tablica as $wiersz)
    {
        echo "<p>$wiersz</>";
    }
```

Pliki – zapis do pliku

Do zapisu służy funkcja – fwrite() Spotykany jest też alias do niej – fputs().

```php
// otwieramy plik w trybie umożliwiającym zapis na końcu pliku
$plik = fopen("plik.txt", "a");

// przypisanie zawartości do zmiennej
$zawartosc = "tekst do zapisania w pliku";
fwrite($plik, $zawartosc);
```

Pliki – zapis do pliku

Dopisywanie tekstu na początku pliku.
Nie ma bezpośredniej możliwości zapisu na początku czy w środku pliku. Można dopisywać tylko na końcu pliku lub zastąpić jego aktualną zawartość.
Jeśli więc chcemy coś zapisać na początku już istniejącego zbioru, należy najpierw otworzyć dany plik, odczytać jego zawartość i dopiero wtedy zapisać całość od nowa.

```php
$plik = fopen("plik.txt", "r");
$dane = fread($plik, filesize("plik.txt"));
fclose($plik);
$noweDane = "Coś do dodania na początku pliku";
$noweDane .= $dane;
$plik = fopen("plik.txt", "w");
fwrite($plik, $noweDane);
fclose($plik);
```

Pliki – blokowanie plików

Podczas używania plików w serwisie o dużej liczbie odwiedzin może się zdarzyć, że w tym samym czasie dwa procesy będą próbowały zapisywać coś do pliku. Może to być przyczyną różnych błędów.

Do zapobiegania takim sytuacji służą blokady plików. Istnieją dwa rodzaje blokad:

- blokada dzielona – może być założona przez wiele procesów naraz podczas odczytu,
- blokada wyłączna – zakładana podczas zapisu do pliku tylko przez jeden proces.

Aby założyć blokadę, należy użyć funkcji flock()

Pobiera ona dwa argumenty:

- uchwyt do pliku
- rodzaj blokady.

Dostępne rodzaje blokady:

LOCK_SH – zakłada blokadę dzieloną (do odczytu),

LOCK_EX – zakłada blokadę wyłączną (do zapisu),

LOCK_UN – zdejmuje blokadę z pliku.

Jeśli zakładanie blokady się powiedzie, to funkcja zwraca wartość true, w przeciwnym wypadku zwracana jest wartość false.

Pliki – blokowanie plików

Przykład blokowania plików:

```
$file = fopen("dane\plik.txt", "w+");
if (flock($file, LOCK_EX)) { // zakładanie blokady wyłącznej
    fwrite($file, "Wartość dopisana do pliku");
    flock($file, LOCK_UN); // zdjęcie blokady
} else {
    echo "Błąd - plik zablokowany";
}
fclose($file)
```

Pliki – informacje o plikach

W PHP istnieje też kilka funkcji, które zwracają informacje o pliku. Należą do nich:

- fileatime(nazwa_pliku) – zwraca datę i czas ostatniego odczytu pliku podane w formacie timestamp (uniksowym),
- filemtime(nazwa_pliku) – zwraca datę i czas ostatniej modyfikacji pliku podane w formacie timestamp,
- filegroup(nazwa_pliku) – zwraca liczbowy identyfikator grupy, do której należy właściciel pliku,
- fileowner(nazwa_pliku) – zwraca identyfikator właściciela pliku,
- fileperms(nazwa_pliku) – zwraca prawa dostępu do pliku,
- filesize(nazwa_pliku) – zwraca wielkość pliku w bajtach.

Oprócz tego istnieje też kilka funkcji zwracających wartości true lub false. Są to:

- is_dir(nazwa_pliku) – informuje o tym, czy zbiór jest katalogiem,
- is_executable(nazwa_pliku) – informuje o tym, czy plik jest wykonywalny,
- is_file(nazwa_pliku) – informuje o tym, że plik istnieje i jest zwykłym plikiem,
- is_readable(nazwa_pliku) – informuje o tym, czy plik można odczytać,
- is_writeable(nazwa_pliku) – informuje o tym, czy plik można zapisywać,
- is_uploaded_file(nazwa_pliku) – informuje o tym, czy plik został wysłany z formularza.

Operacje na plikach

Do kopiowanie plików służy funkcja copy(). Zwraca ona wartość true, jeśli plik zostanie poprawnie skopiowany, lub false, jeżeli wystąpi błąd.

```
if (!copy("tymczasowy/plik.txt", "dane/plik.txt"))
   echo "Błąd podczas kopiowania";
```

Aby zmienić nazwę zbioru, należy użyć funkcji o nazwie rename(). Może ona też służyć do kopiowania plików i zwraca true lub false, tak samo jak w wypadku copy().

```
  if (!rename("/tmp/tymczasowy_plik.txt", "/home/user/login/docs/plik.txt"))
       echo "Błąd podczas kopiowania";
```

Do usuwanie plików służy funkcja o nazwie unlink(). Jako parametr wystarczy podać nazwę pliku.

```
unlink("plik.txt");
```

Operacje na katalogach

Aby utworzyć katalog, należy wywołać funkcję mkdir() i jako to:

- nazwa katalogu do utworzenia
- prawa dostępu (np. 0777).

Funkcja zwraca wartości true lub false w zależności od powodzenia operacji.

```
mkdir("nazwa",0777);
```

Do przeglądania zawartości katalogów służy mechanizm pseudoobiektowy. Przeglądanie zaczynamy funkcją dir(nazwa_folderu), która zwraca obiekt-uchwyt do katalogu. Kolejne pozycje z katalogu pobieramy za pomocą metody read(). Pracę z folderem kończymy metodą close().

```
$dir = dir("logs");
while ($entry = $dir->read()) {
    echo "Kolejna pozycja z tego folderu to $entry<br />";
}
$dir->close();
```

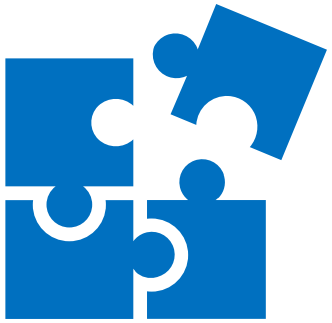
Operacje na katalogach

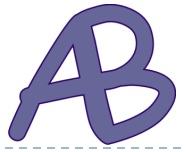
Bardziej skomplikowane jest usuwanie katalogów. Służy do tego funkcja rmdir(),
Jej argumentem jest nazwa folderu przeznaczonego do usunięcia.
Warunkiem koniecznym jest to, aby usuwany katalog był pusty.

Przykład funkcji rekurencyjnej do usuwania całego katalogu (wraz z plikami i podkatalogami):.

```
function deltree($dirn)
{
    $dir = @dir($dirn);
    while ($entry = $dir->read()) {
        if (is_dir($dirn . '/' . $entry) && $entry != '.' && $entry != '..') {
            @deltree($dirn . '/' . $entry);
        } else if ($entry != '.' && $entry != '..') {
            @unlink($dirn . '/' . $entry);
        }
    }
    $dir->close();
    @rmdir($dirn);
}
```

Przykład do wykonania





Literatura

W prezentacji użyto przykładów z książki:

- Żygłowicz Jerzy - PHP - Kompendium wiedzy, Helion

- https://www.php.net