Struktury plików

Tym razem zajmiemy się plikami w PHP, czyli odczytywaniem, zapisywaniem, tworzeniem i usuwaniem, dołączaniem plików do kodu itp.
Nauczymy się również odbierać pliki oraz wysyłać do przeglądarki użytkownika.

PHP include – dołączanie zewnętrznych plików do kodu

Pisanie całego kodu w jednym pliku może być bardzo uciążliwe i czasochłonne. Do tego uniemożliwia korzystanie z ważnej zalety PHP – wielokrotnego użycia napisanych funkcji w różnych plikach. Co to oznacza w praktyce? Mając dwadzieścia podstron nie musimy już w każdej z nich zmieniać stopki czy nagłówka ręcznie! Wystarczy zmienić funkcję odpowiedzialną za wyświetlanie części kodu, a na każdej podstronie zobaczymy zmianę.
Zobaczmy praktyczny przykład użycia. Tworzymy plik **funkcje.php:**

```
<?php

// definicja funkcji stopki
function stopka()
{
  echo „Prawa autorskie: Moja firma 2020”;

}

?>
```

Załóżmy, że tak wygląda nasza stopka w każdej podstronie. Żeby móc korzystać z tak zdefiniowanej funkcji na wszystkich stronach, niezbędne jest umieszczenie jej w pliku, który dołączymy do pozostałych. Plik zawierający funkcję **stopka()** nazwiemy **funkcje.php.** Wystarczy teraz użyć instrukcji **include()** lub **require()** (są niemal równoważne) by można było skorzystać z zadeklarowanej funkcji.
Poniżej listing przykładowego pliku **podstrona1.php**:

```
<?php

include("funkcje.php");

?>

<html>
<head>
  <title>Przykład</title>
</head>
<body>
<?php

  // przykładowa zawartość strony

  stopka(); // zastosowanie funkcji z zewnętrznego pliku

?>
</body>
```

```
</html>
```

PHP require – ten sam efekt, inna obsługa błędów

Include w PHP można stosować zamiennie z **require**. **Require** rzuci wyjątkiem **FATAL_ERROR** w przypadku niepowodzenia, natomiast **include** jedynie wyświetli **Warning**.

Co to oznacza w praktyce?
Require zastopuje wykonywanie skryptu PHP w momencie, gdy nie uda się znaleźć pliku.
Include wykona całość nawet, gdy plik nie zostanie dołączony.
Teraz, w zależności od krytyczności projektu, możesz świadomie zdecydować, z której skorzystasz. Zależy to głównie od tego, jak ważne jest powodzenie podczas dołączenia pliku i czy jest on niezbędny do prawidłowego funkcjonowania reszty kodu.

Include_once oraz Require_once

Istnieją jeszcze drobne modyfikacje powyższych dwóch komend załączających pliki. Noszą nazwy **include_once()** oraz **require_once().** Różnią się od swoich odpowiedników wyłącznie tym, że mimo kilku wywołań w jednym skrypcie, plik zostanie załączony wyłącznie raz, co uchroni nas przed przypadkowym załączeniem jednego pliku kilka razy.

Tworzenie i otwieranie pliku

PHP daje nam możliwość tworzenia plików w dostępnych katalogach. Mogą być to pliki o dowolnym rozszerzeniu, które sami nadajemy. Należy uważać, by nie stworzyć pliku o nazwie, która już istnieje – istniejący plik zostanie wtedy trwale usunięty w miejsce nowo stworzonego. Stwórzmy teraz nowy plik:

```
<?php

// zastosowanie instrukcji fopen
$uchwyt = fopen("style.css", "w");

?>
```

Powyższy listing przedstawia otwarcie pliku za pomocą instrukcji **fopen**. Przyjmuje ona dwa argumenty (właściwie to może aż cztery, ale nas będą interesować wyłącznie te dwa). Pierwszym z nich jest ścieżka do pliku, który chcemy otworzyć. Drugi to tryb, w jakim otwieramy plik. Tryb „w” otwiera plik tylko do zapisu lub tworzy nowy, jeśli ścieżka prowadzi do nieistniejącego pliku.
Jak łatwo się domyślić, żeby utworzyć nowy plik, wystarczy podać ścieżkę do niego prowadzącą, a PHP zrobi resztę.

Usuwanie pliku

Za pomocą PHP możemy również usuwać pliki. Służy do tego funkcja **unlink().** Jako argument przyjmuje ścieżkę do pliku, który ma zostać usunięty. Zobaczmy przykład poniżej:

```
<?php

// zastosowanie instrukcji fopen
```

```
$uchwyt = fopen("style.css", "w");

// usunięcie pliku
unlink("style.css");

?>
```

Nie każdy plik może zostać usunięty. Przede wszystkim musimy być właścicielem tego pliku oraz musi on posiadać odpowiednie prawa dostępu.

Tryby otwarcia pliku

W poprzednim punkcie nauczyliśmy się tworzyć oraz otwierać pliki za pomocą instrukcji **fopen**. Drugim przyjmowanym przez nią parametrem był tryb otwarcia. Poniżej znajdziecie kompletne zestawienie trybów, które można wykorzystać podczas otwierania pliku. Każdy jest krótko opisany. Informacje pochodzą ze strony php.net – fopen
Zestawienie trybów 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 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(2).
‚x+’	Tworzy i otwiera plik odczytu i 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(2).

PHP fgets – czytanie zawartości pliku

Istnieje kilka sposobów na odczytanie zawartości pliku. Zanim skorzystamy z któregokolwiek z nich musimy określić uchwyt do pliku w trybie do odczytu. Listę dostępnych trybów znajdziecie na poprzedniej lekcji.

Fread

Funkcja **fread($uchwyt, $długość)** czyta z pliku określoną długość znaków. Czytanie zakończy się, jeżeli osiągnięta zostanie **$dlugość** znaków, koniec pliku lub odczytanych zostanie 8192 bajtów. Poniżej przykład użycia:

```
<?php

// utworzenie uchwytu do pliku
$plik = fopen('index.html','r');

// przypisanie zawartości do zmiennej
$zawartosc = fread($plik, 8192);

echo $zawartosc;

?>
```

Fgets

Innym sposobem na czytanie zawartości jest funkcja **fgets($uchwyt)**. Czyta ona jedną linię pliku, dopóki nie napotka znacznika przejścia do kolejnej linii. Żeby odczytać cały plik, wystarczy użyć pętli. W przypadku tej metody nie musimy się martwić, czy cały plik zostanie odczytany, czy przypadkiem nie przekroczyliśmy długości znaków. Funkcja **feof()** zwraca **true**, jeśli osiągnęliśmy koniec pliku.
Przykład czytania pliku z użyciem **fgets**:

```
<?php

// utworzenie uchwytu do pliku
$plik = fopen('index.html','r');

$zawartosc = '';

// przypisanie zawartości do zmiennej
while(!feof($plik))
{
  $linia = fgets($plik);
  $zawartosc .= $linia;
}

echo $zawartosc;

?>
```

Fgetc

Fgetc($plik) jest bardzo podobna w użyciu do **fgets**, z tą jednak różnicą, że czytamy po jednym znaku. Przykład niemal identyczny:

```
<?php

// utworzenie uchwytu do pliku
$plik = fopen('index.html','r');

$zawartosc = '';

// przypisanie zawartości do zmiennej
while(!feof($plik))
{
  $linia = fgetc($plik);
  $zawartosc .= $linia;
}

echo $zawartosc;

?>
```

PHP zapis do pliku – zapisywanie zawartości

Aby zapisać dane do pliku używamy funkcji **fwrite()**. Jest ona równoważna z instrukcją **fputs().** W przypadku zapisywania nie potrzebujemy tylu różnych poleceń, jak podczas odczytywania. Cały zapisywany tekst jest umieszczony w jednej zmiennej.

Fwrite

Funkcja **fwrite($uchwyt, $tresc)** zapisuje do pliku tekst, zawarty w zmiennej **$tresc**. Podczas wywołania funkcji można dodać trzeci, opcjonalny argument, **$dlugosc**. Jeżeli go zamieścimy, do pliku zostanie zapisanych maksymalnie **$dlugosc** znaków. Zależnie od trybu otwarcia, zawartość zostanie dopisana na końcu pliku lub nadpisze istniejącą treść.

```
<?php

// utworzenie uchwytu do pliku
// tryb a umożliwia zapis na końcu pliku
$plik = fopen('index.html','a');

// przypisanie zawartości do zmiennej
$zawartosc = "Przykładowa treść, którą umieścimy w pliku.";

fwrite($plik, $zawartosc);

?>
```

Jeżeli chcemy ograniczyć rozmiar docelowego pliku, możemy ustawić opcjonalny argument dla funkcji **fwrite**. Poniżej przykład z użyciem limitu znaków:

```
<?php
```

```
// utworzenie uchwytu do pliku
$plik = fopen('index.html','a');

// przypisanie zawartości do zmiennej
$zawartosc = "Przykładowa treść, którą umieścimy w pliku. ";
$zawartosc .= "Utniemy ciąg po 30 znakach.";

fwrite($plik, $zawartosc, 30);

?>
```

Programując w PHP, często zachodzi potrzeba manipulacji plikami. Czy to odczyt, stworzenie nowego pliku, usunięcie go, czy właśnie zapisanie nowej zawartości. PHP daje nam do dyspozycji cały szereg funkcji, które nam to umożliwiają. Ważne, by je poznać i dobrze zrozumieć.

Podsumowanie

Skrypt, który otworzy dany plik, następnie wyszuka w nim wszystkie adresy e-mail, pasujące do wzorca, po czym zapisze je w pliku na naszym serwerze. Funkcja zostanie napisana w osobnym pliku, po czym załączymy ją za pomocą instrukcji **require_once()**.
Przegląd rozwiązania zaczniemy od pliku **funkcja.php**. Wyrażenie regularne, rozpoznające adres e-mail, zaczerpniemy z poprzedniej lekcji:

```
<?php

function zdobadz_email($strona)
{
  // formuła prawidłowego adresu e-mail
  $sprawdz = '/^[a-zA-Z0-9.\-_]+@[a-zA-Z0-9\-.]+\.[a-zA-Z]{2,4}$/';

  $plik = fopen($strona,'r'); // otwarcie pliku strony
  // utworzenie naszego pliku
  $moj_plik = fopen('tymczasowy_index.txt','a');
  flock($moj_plik, 2); // blokada pliku

  // przeszukujemy plik, dopóki nie znajdziemy się na końcu
  while(!feof($plik))
  {
    $linia = fgets($plik); // pobieramy jedną linię
    // sprawdzamy, czy znajduje się tam adres e-mail
    // jeśli tak, zapisujemy do naszego pliku
    if (preg_match($sprawdz, $linia))
      fputs($moj_plik, $linia);
  }
  fclose($plik); // zamykamy plik strony
}

?>
```

Teraz jeszcze tylko index.php i zabieram się za omawianie poszczególnych kroków:

```
<?php

require_once("funkcja.php");

?>
<html>
<head>
  <title>Przeszukiwacz stron</title>
</head>
<body>
<?php

  zdobadz_email("a.txt");

?>
<p>gotowe</p>
</body>
</html>
```

Przeanalizujmy teraz działanie skryptu. W pierwszej kolejności, do pliku index.php, dołączany jest plik funkcja.php, zawierający napisaną przez nas funkcję, której możemy teraz używać w indexie. Jako argument przyjmuje nazwę pliku. Plik otwieramy przy pomocy funkcji fopen() w trybie ‚r'. Następnie tworzony jest plik tymczasowy, w którym będziemy przechowywać informacje o znalezionych adresach.
Teraz parę słów o blokadzie pliku (funkcja flock()). W przypadku, gdy tylko my korzystamy z pliku, funkcja jest zbędna. Sytuacja zmienia się diametralnie, gdy na naszej stronie panuje duży ruch, a plik modyfikowany jest bardzo często. Może się wtedy zdarzyć, że dwie osoby w tym samym czasie próbują nadpisać plik. Chcemy uniknąć tej sytuacji, dlatego umożliwiamy dostęp do zapisu pliku tylko jednemu użytkownikowi na czas operacji. Gdy skończymy używać pliku, odblokowujemy go.
Plik czytamy po linijce, sprawdzając, czy znajduje się tam ciąg znaków, pasujący do wzorca adresu e-mail. Jeżeli tak, zapisujemy go do naszego pliku tymczasowego. Po skończeniu przeglądania pliku, zamykamy go.

Zadania

Spróbuj swoich sił w tych zadaniach:

- Napisz funkcję, która sprawdzi dany plik pod kątem obecności wulgaryzmów. Jeżeli jakiś znajdzie, zastąpi go łańcuchem znaków -cenzura-.