Praca na plikach

Dołączanie pliku

Możliwe jest wstawienie zawartości jednego pliku PHP do innego pliku PHP (przed wykonaniem go przez serwer), z instrukcją include lub require.

Instrukcje include i require są identyczne, z wyjątkiem niepowodzenia:

- require spowoduje błąd krytyczny (E_COMPILE_ERROR) i zatrzyma skrypt
- include wygeneruje ostrzeżenie (E_WARNING), a skrypt będzie kontynuowany

Tak więc, jeśli chcesz, aby wykonanie było kontynuowane i wyświetlało użytkownikom dane wyjściowe, nawet jeśli brakuje pliku włączającego, użyj instrukcji include. W przeciwnym razie, w przypadku FrameWork, CMS lub złożonego kodowania aplikacji PHP, zawsze używaj instrukcji require, aby dołączyć plik klucza do przepływu wykonywania. Pomoże to uniknąć naruszenia bezpieczeństwa i integralności twojego oprogramowania, wystarczy przypadek, w którym jeden plik klucza zostanie przypadkowo pominięty.

W tym pliki oszczędzają dużo pracy. Oznacza to, że możesz utworzyć standardowy nagłówek, stopkę lub plik menu dla wszystkich swoich stron internetowych. Następnie, gdy nagłówek wymaga aktualizacji, można aktualizować tylko plik nagłówkowy.

Składnia

include ' filename ';

require ' filename ';

Załóżmy, że mamy standardowy plik stopki o nazwie "footer.php", który wygląda następująco:

```
<?php
echo "<p>Copyright &copy; 1999-" . date("Y") . " ZSE</p>";
?>
```

Aby dołączyć plik stopki na stronie, użyj instrukcji include :

```
<?php include 'footer.php';?>
```

Załóżmy, że mamy standardowy plik menu o nazwie "menu.php":

```
<?php
echo '<a href="/default.asp">Home</a> -
<a href="/html/default.asp">HTML Tutorial</a> -
<a href="/css/default.asp">CSS Tutorial</a> -
<a href="/js/default.asp">JavaScript Tutorial</a> -
<a href="default.asp">PHP Tutorial</a>';
?>
```

Wszystkie strony w witrynie internetowej powinny korzystać z tego pliku menu. Oto, jak można to zrobić (używamy elementu , aby menu można było łatwo stylizować później):

```
<div class="menu">
<?php include 'menu.php';?>
```

Instrukcja require jest również używana do dołączenia pliku do kodu PHP.

Istnieje jednak jedna duża różnica między włączeniem i wymaganiem; kiedy plik jest dołączony do instrukcji include , a PHP nie może go znaleźć, skrypt będzie kontynuował wykonywanie:

```
<?php include 'noFileExists.php';
echo "I have a $color $car.";
?>
```

Jeśli zrobimy ten sam przykład za pomocą instrukcji require instrukcja echo nie zostanie wykonana, ponieważ wykonanie skryptu zginie po tym, jak instrukcja require zwróci błąd krytyczny:

```
<?php require 'noFileExists.php';
echo "I have a $color $car.";
?>
```

Obsługa plików

PHP ma kilka funkcji do tworzenia, czytania, przesyłania i edycji plików.

Funkcja PHP **readfile** () odczytuje plik i zapisuje go w buforze wyjściowym.

Załóżmy, że mamy plik tekstowy o nazwie "webdictionary.txt", przechowywany na serwerze, który wygląda następująco:

AJAX = Asynchronous JavaScript and XML
CSS = Cascading Style Sheets
HTML = Hyper Text Markup Language
PHP = PHP Hypertext Preprocessor
SQL = Structured Query Language
SVG = Scalable Vector Graphics
XML = EXtensible Markup Language

Kod PHP do odczytu pliku i zapisania go do bufora wyjściowego jest następujący (funkcja readfile() zwraca liczbę bajtów odczytanych po pomyślnym wykonaniu):

```
echo readfile("webdictionary.txt");
```

Lepszą metodą otwierania plików jest funkcja **fopen**() . Ta funkcja daje więcej opcji niż funkcja readfile() .

Podczas lekcji użyjemy pliku tekstowego "webdictionary.txt":

Pierwszy parametr fopen() zawiera nazwę pliku, który ma zostać otwarty, a drugi parametr określa, w którym trybie plik ma zostać otwarty. Poniższy przykład również generuje komunikat, jeśli funkcja fopen () nie może otworzyć określonego pliku:

```
$myfile = fopen("webdictionary.txt", "r") or die("Unable to open file!");
echo fread($myfile,filesize("webdictionary.txt"));
fclose($myfile);
```

Plik może zostać otwarty w jednym z następujących trybów:

'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(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 okresleniem flag O_EXCL|O_CREAT stosowanym w wywołaniu systemowym open(2).

Funkcja **fread**() czyta z otwartego pliku. Pierwszy parametr fread() zawiera nazwę pliku do odczytania, a drugi parametr określa maksymalną liczbę bajtów do odczytania.

```
fread($myfile,filesize("webdictionary.txt"));
```

Funkcja **fclose**() służy do zamykania otwartego pliku. Dobrą praktyką programowania jest zamykanie wszystkich plików po ich zakończeniu.

Funkcja fclose() wymaga nazwy pliku (lub zmiennej przechowującej nazwę pliku), którą chcemy zamknąć:

```
$myfile = fopen("webdictionary.txt", "r");
// some code to be executed....
fclose($myfile);
```

Odczyt pojedynczej linii PHP - fgets ()

Funkcja **fgets**() służy do odczytu pojedynczej linii z pliku.

```
$myfile = fopen("webdictionary.txt", "r") or die("Unable to open
file!");
echo fgets($myfile);
fclose($myfile);
```

Uwaga: Po wywołaniu funkcji fgets() wskaźnik pliku przeniósł się do następnego wiersza.

Funkcja **feof**() sprawdza, czy osiągnięto "koniec pliku" (EOF). Funkcja feof() jest przydatna do zapętlenia danych o nieznanej długości.

```
$myfile = fopen("webdictionary.txt", "r") or die("Unable to open
file!");
// Output one line until end-of-file
while(!feof($myfile)) {
  echo fgets($myfile) . "<br>";
}
fclose($myfile);
```

Odczyt pojedynczego znaku PHP - **fgetc** ()

Funkcja fgetc() służy do odczytu pojedynczego znaku z pliku. Poniższy przykład odczytuje plik "webdictionary.txt" po znaku, aż do osiągnięcia końca pliku:

```
$myfile = fopen("webdictionary.txt", "r") or die("Unable to open
file!");
// Output one character until end-of-file
while(!feof($myfile)) {
  echo fgetc($myfile);
}
fclose($myfile);
```

Uwaga: Po wywołaniu funkcji fgetc() wskaźnik pliku przechodzi do następnego znaku.

Uprawnienia do plików PHP

Jeśli podczas próby uruchomienia tego kodu występują błędy, sprawdź, czy przyznałeś dostęp do pliku PHP, aby zapisać informacje na dysku twardym.

Funkcja **fwrite**() służy do zapisu do pliku. Pierwszy parametr fwrite() zawiera nazwę pliku do zapisania, a drugi parametr jest napisem do zapisania.

```
$myfile = fopen("newfile.txt", "w") or die("Unable to open
file!");
$txt = "John Doe\n";
fwrite($myfile, $txt);
$txt = "Jane Doe\n";
```

```
fwrite($myfile, $txt);
fclose($myfile);
```

Nadpisywanie PHP

Teraz, gdy "newfile.txt" zawiera pewne dane, możemy pokazać, co się stanie, gdy otworzymy istniejący plik do zapisu. Wszystkie istniejące dane zostaną WYMAZANE i rozpoczynamy od pustego pliku.

```
$myfile = fopen("newfile.txt", "w") or die("Unable to open
file!");
$txt = "Mickey Mouse\n";
fwrite($myfile, $txt);
$txt = "Minnie Mouse\n";
fwrite($myfile, $txt);
fclose($myfile);
```

Jeśli teraz otworzymy plik "newfile.txt", zarówno John, jak i Jane zniknęli, a obecne są tylko dane, które właśnie napisaliśmy:

Przesyłanie plików w PHP 5

Skonfiguruj plik "php.ini"

Najpierw upewnij się, że PHP jest skonfigurowane tak, aby zezwalało na przesyłanie plików.

W pliku "php.ini" wyszukaj dyrektywę file_uploads i ustaw ją na On:

file_uploads = On

```
<form action="upload.php" method="post" enctype="multipart/form-
data">
    Select image to upload:
    <input type="file" name="fileToUpload" id="fileToUpload">
    <input type="submit" value="Upload Image" name="submit">
</form>
```

Niektóre zasady, których należy przestrzegać dla powyższego formularza HTML:

- Upewnij się, że formularz używa metody = "post"
- Formularz wymaga również następującego atrybutu: enctype = "multipart / form-data". Określa typ zawartości, który ma być używany podczas przesyłania formularza
- Atrybut type = "file" znacznika <input> pokazuje pole wejściowe jako kontrolkę file-select, z przyciskiem "Browse" obok kontrolki input
- Powyższy formularz wysyła dane do pliku o nazwie "upload.php",

Utwórz skrypt PHP do przesyłania plików

Plik "upload.php" zawiera kod do przesłania pliku:

```
<?php
$target_dir = "uploads/";
$target_file = $target_dir .
basename($_FILES["fileToUpload"]["name"]);
$uploadOk = 1;
$imageFileType =
strtolower(pathinfo($target_file,PATHINFO_EXTENSION));
// Check if image file is a actual image or fake image
if(isset($_POST["submit"])) {
    $check = getimagesize($_FILES["fileToUpload"]["tmp_name"]);
    if($check !== false) {
        echo "File is an image - " . $check["mime"] . ".";
        $uploadOk = 1;
    } else {
        echo "File is not an image.";
        $uploadOk = 0;
    }
}
?>
```

Wyjaśnienie skryptu PHP:

$ target_dir = "uploads /" - określa katalog, w którym plik ma zostać umieszczony

$ plik_docelowy określa ścieżkę do pliku do przesłania

$ uploadOk = 1 nie jest jeszcze używane (zostanie użyte później)

$ imageFileType przechowuje rozszerzenie pliku (małymi literami)

Następnie sprawdź, czy plik obrazu jest rzeczywistym obrazem lub fałszywym obrazem

Sprawdź, czy plik już istnieje

Teraz możemy dodać pewne ograniczenia.

Najpierw sprawdzimy, czy plik już istnieje w folderze "uploads". Jeśli tak, zostanie wyświetlony komunikat o błędzie, a parametr $ uploadOk ma wartość 0:

```
// Check if file already exists
if (file_exists($target_file)) {
    echo "Sorry, file already exists.";
    $uploadOk = 0;
}
```

Ogranicz rozmiar pliku

Pole do wprowadzania plików w powyższym formularzu HTML nosi nazwę "fileToUpload".

Teraz chcemy sprawdzić rozmiar pliku. Jeśli plik jest większy niż 500 KB, wyświetlany jest komunikat o błędzie, a parametr $ uploadOk ma wartość 0:

```
// Check file size
```

```
if ($_FILES["fileToUpload"]["size"] > 500000) {
    echo "Sorry, your file is too large.";
    $uploadOk = 0;
}
```

Ogranicz typ pliku

Poniższy kod pozwala użytkownikom tylko przesyłać pliki JPG, JPEG, PNG i GIF. Wszystkie pozostałe typy plików wyświetlają komunikat o błędzie przed ustawieniem $ uploadOk na 0:

```
// Allow certain file formats
if($imageFileType != "jpg" && $imageFileType != "png" &&
$imageFileType != "jpeg"
&& $imageFileType != "gif" ) {
    echo "Sorry, only JPG, JPEG, PNG & GIF files are allowed.";
    $uploadOk = 0;
}
```

Kompletny skrypt PHP

```
<?php
$target_dir = "uploads/";
$target_file = $target_dir .
basename($_FILES["fileToUpload"]["name"]);
$uploadOk = 1;
$imageFileType =
strtolower(pathinfo($target_file,PATHINFO_EXTENSION));
// Check if image file is a actual image or fake image
if(isset($_POST["submit"])) {
    $check = getimagesize($_FILES["fileToUpload"]["tmp_name"]);
    if($check !== false) {
        echo "File is an image - " . $check["mime"] . ".";
        $uploadOk = 1;
    } else {
        echo "File is not an image.";
        $uploadOk = 0;
    }
}
// Check if file already exists
if (file_exists($target_file)) {
    echo "Sorry, file already exists.";
    $uploadOk = 0;
}
// Check file size
if ($_FILES["fileToUpload"]["size"] > 500000) {
    echo "Sorry, your file is too large.";
    $uploadOk = 0;
}
// Allow certain file formats
if($imageFileType != "jpg" && $imageFileType != "png" &&
$imageFileType != "jpeg"
&& $imageFileType != "gif" ) {
    echo "Sorry, only JPG, JPEG, PNG & GIF files are allowed.";
```

```
    $uploadOk = 0;
}
// Check if $uploadOk is set to 0 by an error
if ($uploadOk == 0) {
    echo "Sorry, your file was not uploaded.";
// if everything is ok, try to upload file
} else {
    if (move_uploaded_file($_FILES["fileToUpload"]["tmp_name"],
$target_file)) {
        echo "The file ". basename(
$_FILES["fileToUpload"]["name"]). " has been uploaded.";
    } else {
        echo "Sorry, there was an error uploading your file.";
    }
}
?>
```