1.7. Работа с файловой системой

При разработке веб-проектов важной задачей является работа с файловой системой. При этом разработчик должен писать алгоритм как для взаимодействия с файловой системой локального компьютера, так и с файловой системой веб-сервера. Все такие операции, как: загрузка файла с локального компьютера на сервер, копирование файла на сервере из одной директории в другую, создание файла, переименование, удаление, редактирование и другие – реализуется с помощью особых функций в PHP. Рассмотрим эти функции:

- copy($source, $dest) – копирует файл $source в файл с именем $dest. Функция возвращает true в случае успешного завершения, или false, в случае возникновения ошибки.
- move_uploaded_file($filename, $destination) – функция проверяет, является ли файл $filename загруженным на сервер (переданным по протоколу HTTP POST). Если файл действительно загружен на сервер, он будет перемещён в место, указанное в аргументе $destination. В отличие от функции copy() эта функция перемещает файл с локального компьютера на сервер.
- rename($oldname, $newname) – пытается переименовать $oldname в $newname, перенося файл между директориями, если необходимо. Если $newname существует, то он будет перезаписан. При переименовании директории с существующим $newname будет выведено предупреждение.
- fopen ($filename, $mode) – открывает файл $filename на операцию в зависимости от значения $mode. $mode – это строка, которая может содержать одно из следующих значений.
  - r – открыть только для чтения, помещает указатель на начало файла.
  - r+ – открыть для чтения и для записи, помещает указатель на начало файла.
  - w – открыть только для записи, помещает указатель на начало файла и очищает все содержимое файла. Если файл не существует, то создается новый.

- w+ – открыть для чтения и для записи, помещает указатель на начало файла и очищает все содержимое файла. Если файл не существует, создается новый.
- a – открыть только для записи, помещает указатель на конец файла. Если файл не существует, создается новый.
- a+ – открыть для чтения и для записи, помещает указатель на конец файла. Если файл не существует, создается новый.

- fclose($filename) – закрывает файл, который ранее был открыт дескриптором fopen(). Возвращает true, в случае успешного завершения, или false, в случае возникновения ошибки.
- feof($filename) – проверка, не конец ли файла. Возвращает true, если указатель файла указывает на EOF или произошла ошибка, иначе возвращает false.
- fgets($handle, [$length]) – читает строку из файлового указателя $handle. $length – необязательное числовое значение, указывающее размер получаемой строки в байтах.
- fread($stream, [$lenght]) – бинарно-безопасное чтение файла. Читает до $length байт из файлового указателя stream и смещает указатель. Чтение останавливается, как только было достигнуто одно из следующих условий: было прочитано $length байт; достигнут EOF (конец файла).
- fwrite($handle, $string, [$length]) – бинарно-безопасная запись в файл. Строку $string функция записывает в файловый поток $handle. Если передан аргумент $length, запись остановится после того, как $length байтов будут записаны или будет достигнут конец строки $string, смотря на то, что произойдёт раньше.
- file($filename) – читает содержимое файла $filename и помещает его в массив. Возвращает файл в виде массива. Каждый элемент массива соответствует строке файла, с символами новой строки включительно. В случае ошибки file() возвращает false.
- file_get_contents($filename) – читает содержимое файла $filename и помещает его в строку.
- unlink($filename) – удаляет файл $filename.

Рассмотрим простой пример работы с файловой системой на PHP – алгоритм открывает файл newname.txt на чтение, затем считывает из этого файла все строки длиной 3 символа и выводит их на экран браузера. После этого происходит безопасное закрытие файла.

```
<?php
if ($fp=fopen("newname.txt","r"))
echo "Работа функции fopen() произведена успешно !<br>";
do {
     $str = fgets ($fp, 3);
     echo $str, " <br>";
}
while (!feof($fp));
if (!fclose($fp)){
     echo "Функция fclose () выполнила ошибку<br>";
}
else {
     echo "Закрытие файла newname.txt осуществлено успеш-
но<br>";
}
?>
```

Рассмотрим еще одну типичную задачу работы с файлами – чтение из файла и получение данных в виде строки.

```
<?php
     //получает содержимое файла в строку
     $filename = "/usr/local/something.txt";
     $handle = fopen($filename, "r");
     $contents = fread($handle, filesize($filename));
     fclose($handle);
?>
```

Помимо чтения необходимо часто использовать операцию запись в файл. При этом в рассмотренном ниже примере запись в файл происходит первых 7 символов.

```
     <?php
     $h = fopen("my_file.html","a");
     $add_text = "Добавим текст в файл.";
     if (fwrite($h,$add_text,7))
          echo "Добавление текста прошло успешно<br>";
     else
          echo "Произошла ошибка при добавлении данных<br>";
          fclose($h);
     ?>
```

Как уже было описано выше, функция move_uploaded_file предназначена для загрузки файла на сервер. Зачастую при подобных операциях важно ограничивать размер загружаемого файла (ресурсы сервера не бесконечны), а также фильтровать файлы по типу (например, разрешать загружать файлы определенных типов). В следующем примере рассмотрено, как может быть организована загрузка файла на сервер, размером не более 30 000 байт (≈30 Кбайт). На лабораторных работах будет рассмотрен пример с проверкой типов загружаемых файлов.

Пусть имеется файл **index.php**, на котором есть форма для загрузки файлов. Форма должна содержать обязательный атрибут enctype="multipart/form-data" для загрузки файлов.

```
<form enctype="multipart/form-data" action="parse.php" meth-
od="post">
    <input type="hidden" name="MAX_FILE_SIZE" value="30000"
    />
    Загрузить файл:
    <input type="file" name="myfile" /><br>
    <input type="submit" value="Отправить файл" />
</form>
```

Обработчиком формы является файл **parse.php**, который должен получить файл с формы и организовать его загрузку на сервер.

```
    <?
    $uploaddir = 'c:/uploads/'; /*будем сохранять загружаемые
    файлы в эту директорию */
    $destination = $uploaddir . $_FILES['myfile']['name']; /*
    имя файла оставим неизменным */
    print "<pre>";
    if (move_uploaded_file( $_FILES['myfile']['tmp_name'],
    $destination)) {
        /* перемещаем файл из временной папки в выбранную
    директорию для хранения */
        print "Файл успешно загружен <br>"; }
        else { echo "Произошла ошибка при загрузке файла.
    Некоторая отладочная информация:<br>"; print_r($_FILES);
        }
    print "</pre>";
    ?>
```

Помимо работы с файлами, в PHP имеется возможность управления директориями: создание каталога и удаление каталога.

- mkdir($directory, [$permission]) – создает каталог $directory. $permission по умолчанию принимает значение 0777, что означает самые широкие права. Больше информации о правах доступа можно узнать на странице руководства функции chmod().
- rmdir($directory) – удаляет указанную директорию. Директория должна быть пустой и должны иметься необходимые для этого права. При неудачном выполнении будет сгенерирована ошибка уровня E_WARNING.