Работа с файлами

Оглавление

[Файловая система и адресация, примеры на базе разных ОС](#_Toc463355809)

[Подключение файлов с кодом](#_Toc463355810)

[Базовые операции работы с файлами – чтение, запись.](#_Toc463355811)

[Домашнее задание](#_Toc463355812)

[Дополнительные материалы](#_Toc463355813)

Все современные ОС используют файлы для организации своей работы. Философия операционных систем Unix гласит, что все есть файл. Linux, под управлением которой находится большинство хостингов под PHP, добавляет – что не файл, то процесс. Также файлы позволяют крайне удобно структурировать логику исходного кода приложений, что применяется во всех современных PHP-фреймворках. Согласитесь, неудобно было бы хранить километры кода в одном-единственном файле.

По этим причинам работа с файлами являет собой одно из фундаментальных навыков работы с языком.

Файловая система и адресация, примеры на базе разных ОС

Но начать стоит не напрямую с работы с файлами, а о том, как происходит их размещение и поиск их в различных популярных ОС.

Любой файл в системе имеет абсолютный путь, идентифицирующий его местонахождение. В ОС Windows и \*nix системах адресация чуть различается:

|  |  |
| --- | --- |
| Windows | \*nix (Linux, MacOS) |
| На верхнем уровне есть т.н. Системный диски (C, D и т.д.) | Система древовидная, начинается от корня (root, “/”) |
| Разделитель пути – любой слэш (\, /) | Разделитель пути – прямой слэш ( / ) |

В любой ОС у Вашего сайта будет корневая директория, в которой будет лежать вся логика и файлы сайта. Обычно она отличается от корневой директории системы. Так или иначе, веб-сервер, обслуживающий соединения с сайтом, будет видеть только файлы, находящиеся уровнем ниже корневой директории сайта, транслируя переданный URL в адрес конечного файла по установленным правилам. При этом все файлы и директории, которые находятся уровнем выше корневой директории, становятся недоступны всем, кто попробует к ним обратиться. Таким образом, родительская для корневой директория является хорошим местом для хранения файлов, требующих усиленной безопасности (файлы с паролями и ключами, например).

Файлы картинок, PDF, CSS и JS не меняются скриптами и называются статическими. Доступ к ним тоже нужен, но стоит выделять под это отдельные директории, чтобы статика хранилась отдельно от файлов скриптов.

Зная вышеперечисленные особенности, мы можем сформировать неплохую и достаточно безопасную структуру хранения файлов нашего сайта. Определим некую корневую директорию, например, C:/openserver/sites/mysite.com. Расположим директории относительно неё:

* public\_html – директория для обращений веб-сервера
  + img – директория хранения изображений
  + css – стили
  + js – javascript
* engine – основная логика сайта, библиотеки и файлы
* config – файлы конфигурации
* data – директория хранения файлов с некими данными
* templates – файлы для шаблонов страниц

Посторонний пользователь не сможет получить доступ к файлам, которые находятся в lib, поскольку он находится на том же уровне иерархии, что и WWW\_ROOT. Данная структура будет использоваться нами в дальнейшем.

Подключение файлов с кодом

Как мы уже проговорили, хранить весь код в одном файле – плохо. Нам нужно научиться подключать файлы с php кодом внутри логики. Для этого используются следующие команды

* include
* require
* include\_once
* require\_once

По сути, подключение файла с кодом аналогично копированию кода в текущий файл, но без самого копирования. Таким образом, файл с кодом может быть подключен во многих местах, но логика хранится в одном и том же файле, что упрощает изменения логики – достаточно поменять код один раз, и он применится везде.

Разница между include и require заключается в том, что при отсутствии подключаемого файла на диске, Ваша программа просто выведет ошибку об этом и продолжит выполнение. Require не только отобразит ошибку, но и вернёт FATAL\_ERROR – тем самым завершив программу.

Суффикс «\_once» в данных функциях указывает, что подключить файл необходимо только 1 раз, независимо от количества вызовов данной функции. Повторное подключение файла с описанием функций приведёт к ошибке.

Наилучшей практикой является создание единой точки входа в файле index.php. Он уже, в свою очередь, подключает конфигуратор, обрабатывает пришедший URL и решает, что нужо показать пользователю. Веб-сервер все динамические запросы принудительно отправляет к index.php.

Базовые операции работы с файлами – чтение, запись.

Зачастую нам будет необходимо совершать чтение из файла. На уровне PHP нельзя взять файл и поместить его в переменную. Для того, чтобы обмениваться данными с файлом, нужно создать поток. Поток – это набор данных в оперативной памяти, который в нашем случае поступает с устройства. PHP может как читать поток, так и писать в него.

Для того, чтобы прочитать информацию из файла, применим следующий код

|  |
| --- |
| <?php  $file = fopen("file.txt","r");  if(!file)  {  echo("Ошибка открытия файла");  }  ?> |

Что же мы сделали? Функция fopen открывает поток, сохраняя его в переменную $file, задавая тем самым ей тип resource. Второй передаваемый параметр – это тип работы с файлом, адрес которого передан в первом параметре. Он может иметь следующие значения

* r (Открыть файл только для чтения; после открытия указатель файла устанавливается в начало файла);
* r+ (Открыть файл для чтения и записи; после открытия указатель файла устанавливается в начало файла);
* w (Создать новый пустой файл только для записи; если файл с таким именем уже есть вся информация в нем уничтожается);
* w+ (Создать новый пустой файл для чтения записи; если файл с таким именем уже есть вся информация в нем уничтожается);
* a (Открыть файл для дозаписи; данные будут записываться в конец файла);
* a+ (Открыть файл для дозаписи и чтения данных; данные будут записываться в конец файла);
* b (Флаг, указывающий на работу (чтение и запись) с двоичным файлом; указывается только в Windows).

Если файл не существует, то $file будет иметь значение false.

Но пока мы ничего не видим. Для того, чтобы вывести данные из файла, к примеру, на экран, есть несколько способов.

Если мы не знаем, насколько большой объём данных будет считан, то данные читаются побайтово.

|  |
| --- |
| <?php  $file = fopen("file.txt","r");  if(!file){  echo("Ошибка открытия файла");  }  else{  $buffer = '';  while (!feof($file)) {  $buffer .= fread($file, 1);  }  echo $buffer;  fclose($file);  }  ?> |

Здесь функция fread считывает по одному байту из файла и помещает их в буфер. По завершению чтения содержимое буфера выводится на экран, а сам поток закрывается при помощи fclose().

В случаях, когда мы знаем объём данных (к примеру, это файл конфигурации), мы можем просто указать объём данных для чтения в fread.

|  |
| --- |
| <?php  $file = fopen("file.txt","r");  if(!file){  echo("Ошибка открытия файла");  }  else{  $buffer = fread($stream, filesize($file));  echo $buffer;  fclose($stream);  }  ?> |

Необязательно всегда использовать такие большие конструкции для работы с файлами. Есть очень удобная функция file\_get\_contents, функционала которой в большинстве случаев будет хватать.

|  |
| --- |
| <?php  echo file\_get\_contents("file.txt");  ?> |

Теперь научимся писать в файл. Это тоже происходит как при помощи потоков, так и в упрощённом виде – в функции file\_put\_contents.

|  |
| --- |
| <?php  $file = fopen("file.txt","r");  if(!file){  echo("Ошибка открытия файла");  }  else{  $buffer = fread($stream, filesize($file));  echo $buffer;  fclose($stream);  }  ?> |

# Домашнее задание

**Базовый блок**

Создайте галерею фотографий. Она должна состоять всего из одной странички, на которой пользователь видит все картинки в уменьшенном виде и форму для загрузки нового изображения. При клике на фотографию она должна открыться в браузере в новой вкладке. Размер картинок можно ограничивать с помощью свойства width. При загрузке изображения необходимо делать проверку на тип и размер файла.

**Продвинутый блок**

При загрузке изображения на сервер должна создаваться его уменьшенная копия. А на странице index.php должны выводиться именно копии. На реальных сайтах это активно используется для экономии трафика. При клике на уменьшенное изображение в браузере в новой вкладке должен открываться оригинал изображения. Функция изменения размера картинок дана в исходниках. Вам необходимо суметь встроить её в систему.

Дополнительные материалы

Литература

* Котеров Д.: PHP 5 в подлиннике
* Head First PHP and MySQL

Дополнительные материалы

* https://ru.wikipedia.org/wiki/Цикл\_(программирование)
* http://php.net/manual/ru/reserved.variables.globals.php