Урок 4

Работа с файлами

Основы работы с файлами и файловыми системами в PHP. Реализация функционала логирования и сохранения информации.

Все современные ОС используют файлы для организации своей работы. Философия операционных систем Unix гласит: «Все есть файл». Linux, под управлением которой находится большинство хостингов под PHP, добавляет: «Что не файл, то – процесс».

Файлы позволяют удобно структурировать логику исходного кода приложений, что применяется во всех современных PHP-фреймворках. Согласитесь, хранить километры кода в одном-единственном файле – сомнительное удовольствие.

Поэтому работа с файлами – один из фундаментальных навыков при освоении языка программирования.

Файловая система и адресация. Примеры на базе разных ОС

Сначала выясним, как происходит размещение и поиск файлов в популярных ОС.

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

Windows	*nix (Linux, MacOS)
На верхнем уровне есть системные 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 расположится вся логика и файлы веб-ресурса. 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. Он транслирует 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. Директория, родительская для корневой, – хорошее место 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 изолированно от файлов скриптов.

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

- **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**. Данную структуру будем использовать в дальнейшем.

Подключение файлов с кодом

Научимся подключать файлы с php-кодом внутри логики. Для этого используются следующие команды:

- **include**;
- **require**;
- **include_once**;
- **require_once**.

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. Это упрощает внесение изменений в логику – достаточно поменять код один раз, и он будет применен везде.

В чем разница между **include** и **require**? Если на диске отсутствует подключаемый файл, команда **include** выведет ошибку и продолжит выполнение программы. **Require** не только отобразит ошибку, но и вернет **FATAL_ERROR** – тем самым завершив программу.

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

Лучшая практика – создавать единую точку входа в файле **index.php**. Он подключает конфигуратор, обрабатывает пришедший URL и решает, что нужно показать пользователю. Веб-сервер все динамические запросы принудительно отправляет к **index.php**.

Базовые операции работы с файлами: чтение, запись

Рассмотрим операцию чтения из файла. На уровне PHP нельзя взять файл и поместить его в переменную. 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 выполнять запись в него.

Чтобы прочитать информацию из файла, применим следующий код:

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

Функция **fopen** открывает поток, сохраняя его в переменную **$file**, задавая тем самым ей тип **resource**. Второй 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, то вся информация в нем уничтожается;
- **w+** – создать новый пустой файл для чтения и записи. Если файл с таким именем уже есть, то вся информация в нем уничтожается;
- **a** – открыть файл для дозаписи; данные будут записываться в конец файла;
- **a+** – открыть файл для дозаписи и чтения. 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**.

Чтобы вывести данные из файла на экран, есть несколько способов:

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

```php
<?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 завершении чтения содержимое буфера выводится на экран, а сам поток закрывается при помощи **fclose()**.

Когда мы знаем объем данных (к примеру, это файл конфигурации), то можем просто указать объем 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);
}
?>
```

Не обязательно всегда использовать такие большие конструкции для работы с файлами. Есть очень удобная функция **file_get_contents**, функционала которой в большинстве случаев будет достаточно.

```
<?php
echo file_get_contents("file.txt");
?>
```

Рассмотрим операцию записи в файл. Она выполняется как при помощи потоков, так и в упрощенном виде – в функции **file_put_contents**.

```
<?php
$filename="file.txt";
file_put_contents("file.txt", "Some Data");
?>
```

Практическое задание

1. Создать галерею фотографий. Она должна состоять из одной страницы, на которой пользователь видит все картинки в уменьшенном виде. При клике на фотографию она должна открыться в браузере в новой вкладке. Размер картинок можно ограничивать с помощью свойства **width**.

2. *Строить фотогалерею, не указывая статичные ссылки к файлам, а просто передавая в функцию построения адрес папки с изображениями. Функция сама должна считать список файлов и построить фотогалерею со ссылками в ней.

3. *[для тех, кто изучал JS-1] При клике по миниатюре нужно показывать полноразмерное изображение в модальном окне (материал в помощь: https://www.internet-technologies.ru/articles/sozdaem-prostoe-modalnoe-okno-s-pomoschyu-jquery.html)

Дополнительные материалы

1. https://ru.wikipedia.org/wiki/Цикл_(программирование);
2. http://php.net/manual/ru/reserved.variables.globals.php.

Используемая литература

Для подготовки данного методического пособия были использованы следующие ресурсы:

1. Д. В. Котеров, А. Ф. Костарев. PHP 5 (в подлиннике) — 2-е изд. — СПб, ,2008..
2. Lynn Beighley, Michael Morrison. Head First PHP and MySQL — "O'Reilly Media, Inc.", 2009