УНИВЕРСИТЕТ СИНЕРГИЯ

Работа с файлами. Практика

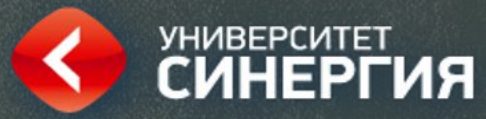
Преподаватель – Борисова Татьяна Андреевна

Сегодня на занятии:

УНИВЕРСИТЕТ СИНЕРГИЯ

1) Отработка практических навыков в работе с файлами;
2) Обсуждение пройденного материала и домашнего задания;

Работа с файлами



Прежде всего следует упомянуть об именах файлов. Если создается код, который может использоваться на различных установках PHP, то узнать о том, чувствительна система к регистру букв или нет, практически невозможно. Например, имена файлов в Windows и Mac OS X нечувствительны к регистру, а в Linux и UNIX — чувствительны. Поэтому нужно принять за основу то, что система чувствительна к регистру, и придерживаться соглашения о присваивании файлам имен в нижнем регистре.

Проверка существования файла

Чтобы проверить факт существования файла, можно воспользоваться функцией file_exists, которая возвращает либо TRUE, либо FALSE и используется следующим образом:

```
if (file_exists("testfile.txt")) echo "Файл существует";
```

Создание файла

УНИВЕРСИТЕТ СИНЕРГИЯ

В данный момент файла testfile.txt не существует, поэтому создадим его и запишем в него несколько строк. Наберите код, показанный в примере 1, и сохраните его под именем testfile.php.

Пример 1 Создание простого текстового файла

```
<?php // testfile.php

$fh = fopen("testfile.txt", 'w') or die("Создать файл не удалось");

$text = <<<_END Строка 1 Строка 2 Строка 3

_END;

fwrite($fh, $text) or die("Сбой записи файла");

fclose($fh);

echo "Файл 'testfile.txt' записан успешно ";

?>
```

Создание файла

Если этот код будет запущен через браузер, то при его успешном выполнении появится следующее сообщение: «Файл 'testfile.txt' записан успешно». Если будет выведено сообщение об ошибке, значит, на диске недостаточно свободного места или, что более вероятно, отсутствует разрешение на создание файла или на запись в этот файл. В таком случае нужно изменить атрибуты папки назначения в соответствии с требованиями вашей операционной системы. Если все обойдётся без ошибки, то файл testfile.txt попадет в ту же папку, где был сохранен программный файл testfile.php. Если открыть файл в текстовом или программном редакторе, в нем будет следующее содержимое:

Строка 1

Строка 2

Строка 3

Создание файла

В этом простом примере показана последовательность работы со всеми файлами.

1. Все начинается с открытия файла с помощью вызова функции fopen.
2. После этого можно вызывать другие функции. В данном случае в файл велась запись (fwrite), но можно также читать данные из уже существующего файла (fread или fgets) и осуществлять с ним другие действия.
3. Работа завершается закрытием файла (fclose). Хотя программа перед завершением своей работы делает это за вас, но все же вы должны удостовериться в том, что по окончании работы с файлом он будет закрыт.

Создание файла

Каждому открытому файлу требуется файловый ресурс, чтобы PHP-программа могла к нему обращаться и им управлять. В предыдущем примере переменной $fh (которую я выбрал в качестве описателя файла) присваивается значение, возвращаемое функцией fopen. После этого каждой функции обработки файла, которая получает к нему доступ, например fwrite или fclose, в качестве параметра должна быть передана переменная $fh, чтобы идентифицировать обрабатываемый файл. Интересоваться содержимым переменной $fh не стоит, это всего лишь номер, используемый PHP для ссылки на внутреннюю информацию о файле. Данная переменная используется только для передачи другим функциям.

Создание файла

В случае сбоя функция fopen возвращает значение FALSE. В предыдущем примере показан простой способ перехвата управления и реакции на сбой: в нем вызывается функция die, которая завершает программу и выдает пользователю сообщение об ошибке. Это упрощённый способ выхода подходит лишь для наших учебных программ, а выходить с его помощью из веб-приложения не следует ни в коем случае (вместо этого нужно создать веб-страницу с сообщением об ошибке).

Обратите внимание на второй параметр, используемый в вызове функции fopen. Это символ w, предписывающий функции открыть файл для записи. Если такого файла нет, то он будет создан. Применять эту функцию следует с оглядкой: если файл уже существует, то параметр режима работы w заставит функцию fopen удалить все его прежнее содержимое (даже если в него не будет записано ничего нового!).

Режимы работы, поддерживаемые функцией fopen

УНИВЕРСИТЕТ СИНЕРГИЯ

Режим	Действие	Описание
'r'	Чтение с начала файла	Открытие файла только для чтения; установка указателя файла на его начало. Возвращение FALSE, если файла не существует
'r+'	Чтение с начала файла с возможностью записи	Открытие файла для чтения и записи; установка указателя файла на его начало. Возвращение FALSE, если файла не существует
'w'	Запись с начала файла с усечением его размера	Открытие файла только для записи; установка указателя файла на его начало и сокращение размера файла до нуля. Если файла не существует, попытка его создания
'w+'	Запись с начала файла с усечением его размера и возможностью чтения	Открытие файла для чтения и записи; установка указателя файла на его начало и сокращение его размера до нуля. Если файла не существует, попытка его создания
'a'	Добавление к концу файла	Открытие файла только для записи; установка указателя файла на его конец. Если файла не существует, попытка его создания
'a+'	Добавление к концу файла с возможностью чтения	Открытие файла для чтения и записи; установка указателя файла на его конец. Если файла не существует, попытка его создания

Чтение из файлов

Проще всего прочитать текстовый файл, извлекая из него всю строку целиком, для чего, как в примере 2, используется функция fgets (последняя буква s в названии функции означает string — «строка»).

Пример 2 Чтение файла с помощью функции fgets

```
<?php

$fh = fopen("testfile.txt", 'r') or die("Файл не существует, или вы не обладаете правами на его открытие");

$line = fgets($fh);

fclose($fh);

echo $line;

?>
```

Чтение из файлов

Если используется файл, созданный кодом из примера 1, будет получена первая строка:

Строка 1

Можно также извлечь из файла сразу несколько строк или фрагменты строк, воспользовавшись функцией fread, как показано в примере 3.

Пример 3 Чтение файла с помощью функции fread

```
<?php

 $fh = fopen("testfile.txt", 'r') or die("Файл не существует, или вы не обладаете правами на его открытие");

$text = fread($fh, 3); fclose($fh); echo $text;

?>
```

Чтение из файлов

При вызове функции fread было запрошено чтение трех символов, поэтому программа отобразит следующий текст:

Стр

Функция fread обычно применяется для чтения двоичных данных. Но если она используется для чтения текстовых данных объемом более одной строки, следует брать в расчет символы новой строки.

Копирование файлов

Попробуем создать клон нашего файла testfile.txt, воспользовавшись PHP-функцией copy. Наберите текст примера 4 и сохраните его в файле copyfile.php, а затем вызовите программу через браузер.

Пример 4 Копирование файла

```
<?php // copyfile.php

copy('testfile.txt', 'testfile2.txt') or die("Копирование невозможно");

echo "Файл успешно скопирован в 'testfile2.txt'»;

?>
```

Копирование файлов

Если еще раз проверить содержимое вашей папки, в ней окажется новый файл testfile2.txt. Кстати, если вам не нужно, чтобы программа завершала свою работу после неудачной попытки копирования, можно воспользоваться другим вариантом синтаксиса, который показан в примере 5

Пример 5 Альтернативный синтаксис для копирования файла

```
<?php // copyfile2.php

if (!copy('testfile.txt', 'testfile2.txt')) echo " Копирование невозможно";

else echo "Файл успешно скопирован в 'testfile2.txt'»;

?>
```

Перемещение файла

Для перемещения файла его следует переименовать, как показано в примере 6.

Пример 6. Перемещение файла

```
<?php // movefile.php

if (!rename('testfile2.txt', 'testfile2.new')) echo "Переименование невозможно";

else echo "Файл успешно переименован в 'testfile2.new'";

?>
```

Функцию переименования можно применять и к каталогам. Чтобы избежать предупреждений при отсутствии исходных файлов, сначала для проверки факта их существования можно вызвать функцию file_exists.

Удаление файла

Для удаления файла из файловой системы достаточно, как показано в примере 7, воспользоваться функцией unlink, позволяющей сделать это.

Пример 7 Удаление файла

```
<?php // deletefile.php

if (!unlink('testfile2.new')) echo "Удаление невозможно ";

else echo "Файл 'testfile2.new' удален успешно";

?>
```

Удаление файла

При непосредственном доступе к файлам на жестком диске нужна гарантия того, что ваша файловая система не будет поставлена под угрозу. Например, при удалении файла на основе введённой пользователем информации нужно быть абсолютно уверенным в том, что этот файл может быть удален без ущерба безопасности системы и что пользователю разрешено удалять его.

Примечание: В данном случае, как и при операции перемещения, если файла с таким именем не существует, будет выведено предупреждение, появления которого можно избежать, если использовать функцию file_exists для проверки его существования перед вызовом функции unlink.

Обновление файлов

Довольно часто возникает потребность добавлять к сохраненному файлу дополнительные данные, для чего существует множество способов. Можно воспользоваться одним из режимов добавления данных или же задействовать режим, поддерживающий запись, и просто открыть файл для чтения и записи и переместить указатель файла в то место, с которого необходимо вести запись в файл или чтение из файла.

Указатель файла — это позиция внутри файла, с которой будет осуществлен очередной доступ к файлу при чтении или записи. Его не следует путать с *описателем файла* (который в примере 1 хранился в переменной $fh), содержащим сведения о том файле, к которому осуществляется доступ.

Обновление файлов

Если набрать код, показанный в примере 8, сохранить его в файле update.php, а затем вызвать его из своего браузера, то можно увидеть работу указателя.

Пример 8 Обновление файла

```
<?php // update.php

$fh = fopen("testfile.txt", 'r+') or die("Сбой открытия файла");

 $text = fgets($fh);

При непосредственном доступе к файлам на жестком диске нужна гарантия того, что ваша файловая система не будет поставлена под угрозу. Например, при удалении файла на основе введённой пользователем информации нужно быть абсолютно уверенным в том, что этот файл может быть удален без ущерба безопасности системы и что пользователю раз- решено удалять его.

fseek($fh, 0, SEEK_END);

fwrite($fh, "$text") or die("Сбой записи в файл");

fclose($fh); echo "Файл 'testfile.txt' успешно обновлен";

?>
```

Обновление файлов

Эта программа открывает файл testfile.txt для чтения и записи, для чего указывается режим работы '+r', в котором указатель устанавливается в самое начало файла. Затем используется функция fgets, с помощью которой из файла считывается одна строка (до встречи первого символа перевода строки). После этого вызывается функция fseek, чтобы переместить указатель файла в самый конец, куда затем добавляется строка, которая была извлечена из начала файла (и сохранена в переменной $text), после чего файл закрывается. Получившийся в итоге файл имеет следующий вид:

Строка 1

Строка 2

Строка 3

Строка 1

Обновление файлов

Первая строка была успешно скопирована, а затем добавлена в конец файла.

В данном примере функции fseek, кроме описателя файла $fh, были переданы еще два параметра — 0 и SEEK_END. Параметр SEEK_END предписывает функции переместить указатель файла в его конец, а параметр 0 показывает, на сколько позиций нужно вернуться назад из этой позиции. В примере 8 используется значение 0, потому что указатель должен оставаться в конце файла.

С функцией fseek можно задействовать еще два режима установки указателя: SEEK_SET и SEEK_CUR. Режим SEEK_SET предписывает функции установку указателя файла на конкретную позицию, заданную предыдущим параметром. Поэтому в следующем примере указатель файла перемещается на позицию 18:

```
fseek($fh, 18, SEEK_SET);
```

Обновление файлов

Режим SEEK_CUR приводит к установке указателя файла на позицию, которая смещена от текущей позиции на заданное значение. Если в данный момент указатель файла находится на позиции 18, то следующий вызов функции переместит его на позицию 23:

```
fseek($fh, 5, SEEK_CUR);
```

Делать это без особой надобности не рекомендуется, но таким образом даже текстовые файлы (с фиксированной длиной строк) можно использовать в качестве простых неструктурированных баз данных. В этом случае ваша программа может использовать функцию fseek для перемещения в обе стороны по такому файлу для извлечения, обновления существующих и добавления новых записей. Записи также могут удаляться путем их перезаписи нулевыми символами и т. д.

Блокирование файлов при коллективном доступе

УНИВЕРСИТЕТ СИНЕРГИЯ

Веб-программы довольно часто вызываются многими пользователями в одно и то же время. Когда одновременно предпринимается попытка записи в файл более чем одним пользователем, файл может быть поврежден. А когда один пользователь ведет в него запись, а другой считывает из него данные, с файлом ничего не случится, но читающий может получить весьма странные результаты.

Чтобы обслужить сразу несколько одновременно обращающихся к файлу пользователей, нужно воспользоваться функцией блокировки файла flock. Эта функция ставит в очередь все другие запросы на доступ к файлу до тех пор, пока ваша программа не снимет блокировку. Когда ваши программы обращаются к файлу, который может быть одновременно доступен нескольким пользователям, с намерением произвести в него запись, к коду нужно также добавлять задание на блокировку файла, как в примере 9, который является обновленной версией примера 8.

Блокирование файлов при коллективном доступе

Пример 9. Обновление файла с использованием блокировки

```
<?php
$fh = fopen("testfile.txt", 'r+') or die("Сбой открытия файла");
$text = fgets($fh);
If (flock($fh, LOCK_EX)) {
fseek($fh, 0, SEEK_END);
fwrite($fh, "$text") or die("Сбой записи в файл");
flock($fh, LOCK_UN);
}
 fclose($fh); echo "Файл 'testfile.txt' успешно обновлен";
?>
```

Блокирование файлов при коллективном доступе

При блокировке файла для посетителей вашего сайта нужно добиться наименьшего времени отклика: блокировку следует ставить непосредственно перед внесением изменений в файл и снимать ее сразу же после их внесения. Блокировка файла на более длительный период приведет к неоправданному замедлению работы приложения. Поэтому в примере 9 функция flock вызывается непосредственно до и после вызова функции fwrite.

При первом вызове flock с помощью параметра LOCK_EX устанавливается эксклюзивная блокировка того файла, ссылка на который содержится в переменной $fh:

```
flock($fh, LOCK_EX);
```

Блокирование файлов при коллективном доступе

УНИВЕРСИТЕТ СИНЕРГИЯ

С этого момента и далее никакой другой процесс не может осуществлять не только запись, но даже чтение файла до тех пор, пока блокировка не будет снята с помощью передачи функции параметра LOCK_UN:

flock($fh, LOCK_UN);

Как только блокировка будет снята, другие процессы снова получат возможность доступа к файлу. Это одна из причин, по которой необходимо заново обращаться к нужному месту в файле при каждом чтении или записи данных: со времени последнего обращения к нему другой процесс мог внести в этот файл изменения.

Блокирование файлов при коллективном доступе

Кстати, вы заметили, что вызов с требованием эксклюзивной блокировки вложен в структуру инструкции if? Дело в том, что flock поддерживается не на всех системах, и поэтому есть смысл проверить успешность установки блокировки, так как известно, что некоторые системы на это не способны.

Следует также принять во внимание, что действия функции flock относятся к так называемой *рекомендательной* блокировке. Это означает, что блокируются только те процессы, которые вызывают эту функцию. Если есть код, который действует напрямую и изменяет файлы, не блокируя их с помощью flock, он всегда сможет обойти блокировку и внести хаос в ваши файлы.

Кстати, если в каком-то кодовом фрагменте заблокировать файл, а затем по рассеянности забыть его разблокировать, это может привести к ошибке, которую будет очень трудно обнаружить.

Чтение всего файла целиком

Для чтения целиком всего файла без использования описателей файлов можно воспользоваться очень удобной функцией file_get_contents. Она очень проста в применении, о чем свидетельствует код примера 10.

Пример 10 Использование функции file_get_contents

```
<?php

echo "<pre>"; // Тег, позволяющий отображать переводы строк

echo file_get_contents("testfile.txt");

echo "</pre>"; // Прекращение действия тега pre

?>
```

Чтение всего файла целиком

Но эту функцию можно использовать и с большей пользой. С ее помощью можно извлечь файл с сервера через Интернет. В примере 11 показан запрос кода HTML с главной страницы сайта O'Reilly с последующим ее отображением, как при обычном переходе на саму веб-страницу.

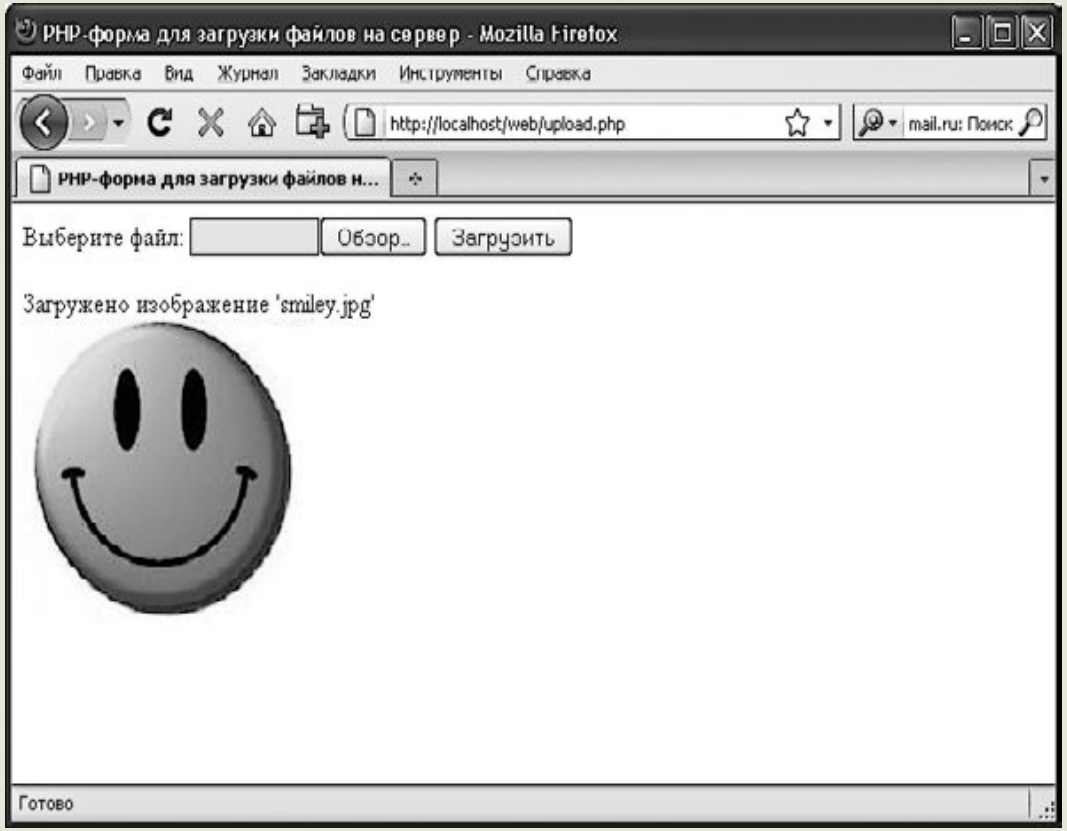
Пример 11 Захват главной страницы сайта O'Reilly

```
<?php

echo file_get_contents("http://oreilly.com");

?>
```

Использование массива $_FILES



При загрузке файла на сервер в массиве $_FILES сохраняются пять элементов, показанных в таблице на следующем слайде (где используется загружаемый файл, имя которого предоставляется отправляемой серверу формой).

Содержимое массива $_FILES

Элемент массива	Содержимое
$_FILES['file']['name']	Имя загруженного файла (например, smiley.jpg)
$_FILES['file']['type']	Тип содержимого файла (например, image/jpeg)
$_FILES['file']['size']	Размер файла в байтах
$_FILES['file']['tmp_name']	Имя временного файла, сохраненного на сервере
$_FILES['file']['error']	Код ошибки, получаемый после загрузки файла

Чтение всего файла целиком

Типы содержимого обычно называли MIME-типами (Multipurpose Internet Mail Extension — многоцелевые почтовые расширения в Интернете). Но поскольку позже они были распространены на все виды передаваемой через Интернет информации, то теперь их часто называют типами информации, используемой в Интернете (Internet media types). В табл. 7.7 показаны некоторые из наиболее часто используемых типов, появляющиеся в элементе массива $_FILES['file']['type'].

application/pdf	image/gif	multipart/form-data	text/xml
application/zip	image/jpeg	text/css	video/mpeg
audio/mpeg	image/png	text/html	video/mp4
audio/x-wav	image/tiff	text/plain	video/quicktime

Домашнее задание

1. Создать файл только для чтения.
2. Проверить, существует ли он теперь.
3. Сделать копию и переместить его.
4. Затем удалить первый созданный файл, оставив лишь его копию.

СПАСИБО ЗА ВНИМАНИЕ!