Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет

информатики и радиоэлектроники»

Факультет информационных технологий и управления

Кафедра информационных технологий автоматизированных систем

Отчет по лабораторной работе №6 по дисциплине

ТЕХНОЛОГИИ ИНТЕРНЕТ-ПРОГРАММИРОВАНИЯ

на тему

«Работа с файлами»

|  |  |
| --- | --- |
| Выполнил ст. гр. 820601  Проверил преп. каф. ИТАС | А.Р. Шведов  А.Л. Гончаревич |

Минск 2022

**СОДЕРЖАНИЕ**

[Введение 3](#_Toc96283290)

[1 Постановка задачи 4](#_Toc96283291)

[2 Теоретическая часть 5](#_Toc96283292)

[2.1 Файлы в *PHP* 5](#_Toc96283293)

[2.2 Используемые в работе функции, методы и объекты 5](#_Toc96283294)

[3 Ход работы 7](#_Toc96283295)

[3.1 Постановка задачи 7](#_Toc96283296)

[3.2 Техническое выполнение 7](#_Toc96283297)

[3.3 Руководство пользователя 10](#_Toc96283298)

[Заключение 13](#_Toc96283299)

Введение

*PHP* — язык программирования, который наиболее распространён в сфере веб-разработки. Язык *PHP* работает на удаленном сервере, поэтому он и называется серверный язык программирования.

Любой скрипт *PHP* состоит из последовательности операторов. Оператор может быть присваиванием, вызовом функции, циклом, условным выражением или пустым выражением. Операторы обычно заканчиваются точкой с запятой. Также операторы могут быть объединены в группу заключением группы операторов в фигурные скобки. Группа операторов также является оператором.

Интернет — это множество компьютеров по всему миру, соединённых между собой проводами в единую сеть. Все компьютеры делятся на две большие группы: клиенты и сервера. Клиенты инициируют запросы на сервера, а те, в свою очередь, их принимают, обрабатывают и отправляют клиенту ответ.

*PHP* позволяет решить множество задач связанных с клиент-серверной архитектурой, например:

1 С помощью *HTML* можно только создать форму. А обработать то, что ввёл пользователь, может лишь *PHP*.

2 Если делать блог на на чистом *HTML*, то на каждую статью требуется создавать новый файл. Добавлять и редактировать записи придётся вручную. PHP позволяет обойтись с помощью одного файла, а статьи хранить в базе данных. Благодаря этому, можно сделать админку, из которой можно будет добавлять и редактировать контент.

3 *PHP* позволяет реализовать механизм авторизации на сайте.

1. Постановка задачи

Изучить семантику, синтаксис и возможности языка *PHP.* Изучение базового синтаксиса в языке *PHP*, работа с фалами: чтение, запись, добавление информации. Ознакомление с основными функциями *PHP* для работы с файлами.

1. Теоретическая часть
   1. Файлы в *PHP*

*File* — это именованная область данных на носителе информации, используемая как базовый объект взаимодействия с данными в операционных системах.

В *PHP* существует два режима работы с файлами: текстовый и бинарный. Первый используется для работы с текстовыми документами, а второй применяется для операций с байтами информации абсолютно любого файла. Однако, поскольку в *PHP* нет типа данных «байт», работа всегда ведётся со строковыми данными. Поэтому разница между текстовым и бинарным режимами практически отсутствует. Она заключается лишь в том, что в системах семейства *Unix* для перевода строки используется символ «*\n*», а в *Windows* – последовательность «*r\n*». При работе в текстовом режиме *PHP*-интерпретатор сам определит, какой вариант нужно использовать.

Согласно документации в *PHP* выделяют следующие виды режима открытия файлов:

*1 «r»* – открытие файла только для чтения.

2 «*r*+» – открытие файла одновременно на чтение и запись.

*3 «w»* – создание нового пустого файла. Если на момент вызова уже существует такой файл, то он уничтожается.

*4 «w*+» – аналогичен «*r*+», только если на момент вызова фай такой существует, его содержимое удаляется.

*5 «а»* – открывает существующий файл в режиме записи, при этом указатель сдвигается на последний байт файла (на конец файла).

6 «*a*+» – открывает файл в режиме чтения и записи при этом указатель сдвигается на последний байт файла (на конец файла). Содержимое файла не удаляется.

* 1. Используемые в работе функции, методы и объекты

*MIME (Multipurpose Internet Mail Extension,* Многоцелевые расширения почты Интернета*) —* спецификация для передачи по сети файлов различного типа: изображений, музыки, текстов, видео, архивов и др. Указание *MIME*-типа используется в *HTML* обычно при передаче данных форм и вставки на страницу различных объектов. Например, *image/jpeg.*

В работе используются следующие функции:

*1 finfo* — класс, который предоставляет объектно-ориентированный интерфейс к функциям *fileinfo*.

*2 finfo*\_open() — функция, открывающая файл в как «магическую базу данных» и возвращая ее экземпляр.

*3 finfo\_close()* — функция, которая закрывает экземпляр *finfo.*

*4 getimagesize($file)* — определит размер любого заданного, поддерживаемого изображения и вернёт этот размер вместе с типом файла и текстовой строкой *height/width*, которую можно будет использовать внутри тега *HTML IMG*, а также вернёт соответствующий тип содержимого *HTTP*.

*5 imagecreatetruecolor()* — возвращает объект, представляющий чёрное изображение заданного размера.

*6 imagecopyresampled()* — копирует прямоугольную часть одного изображения на другое изображение, интерполируя значения пикселов таким образом, чтобы уменьшение размера изображения не уменьшало его чёткости.

*7 move\_uploaded\_file($from, $to)* — проверяет, является ли файл *from* загруженным на сервер (переданным по протоколу *HTTP POST*). Если файл действительно загружен на сервер, он будет перемещён в место, указанное в аргументе *to*.

1. Ход работы
   1. Постановка задачи

Работа выполняется с помощью редактора кода – *Visual Studio Code*. Работа с программой начинается с создания *PHP* файла, в который будут помещаться скрипты.

Согласно заданию необходимо создать 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.

При загрузке изображения необходимо делать проверку на тип и размер файла. ри 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.

Приведём структуру проекта на рисунке 3.1.

Text

Description automatically generated with medium confidence

Рисунок 3.1 — Структура проекта

* 1. Техническое выполнение

Приведём основной фрагмент кода, выполняющий отображение картинок на странице *index.php*. Заметим, что сохраненные в директории *smaller/* изображения уже правильного размера, поэтому при выводе мы их не ограничиваем через тэги *HTML*.

<?php

$folder\_path = 'img/smaller/'; //image's folder path

$num\_files = glob($folder\_path . "\*.{JPG,jpg,gif,png,bmp}", GLOB\_BRACE);

$folder = opendir($folder\_path);

if ($num\_files > 0) {

while (false !== ($file = readdir($folder))) {

$file\_path = $folder\_path . $file;

$orig\_path = "img/original/" . $file;

$extension = strtolower(pathinfo($file, PATHINFO\_EXTENSION));

if ($extension == 'jpg' || $extension == 'png' || $extension == 'jpeg') {

?>

<a target="\_blank" href="<?php echo $orig\_path; ?>"><img src="<?php echo $file\_path; ?>" /> </a>

<?php

}

}

} else {

echo "the folder was empty !";

}

closedir($folder);

?>

Приведём фрагмент кода, который отвечает за форму загрузки файлов:

<body>

<?php

$folder\_path = 'img/smaller/'; //image's folder path

$num\_files = glob($folder\_path . "\*.{JPG,jpg,gif,png,bmp}", GLOB\_BRACE);

$folder = opendir($folder\_path);

if ($num\_files > 0) {

while (false !== ($file = readdir($folder))) {

$file\_path = $folder\_path . $file;

$orig\_path = "img/original/" . $file;

$extension = strtolower(pathinfo($file, PATHINFO\_EXTENSION));

if ($extension == 'jpg' || $extension == 'png' || $extension == 'jpeg') {

?>

<a target="\_blank" href="<?php echo $orig\_path; ?>"><img src="<?php echo $file\_path; ?>" /> </a>

<?php

}

}

} else {

echo "the folder was empty !";

}

closedir($folder);

?>

<form action="upload.php" method="post" enctype="multipart/form-data">

Select image to upload:

<input type="file" accept="image/\*" name="fileToUpload" id="fileToUpload" required>

<input type="submit" value="Upload Image" name="submit">

</form>

</body>

Приведём фрагмент кода, который отвечает за копирование изображение в его уменьшенную копию:

function resize($originalFile, $targetFile, $newwidth, $newheight) {

$info = getimagesize($originalFile);

$mime = $info['mime'];

switch ($mime) {

case 'image/jpeg':

$image\_create\_func = 'imagecreatefromjpeg';

$image\_save\_func = 'imagejpeg';

break;

case 'image/jpg':

$image\_create\_func = 'imagecreatefromjpeg';

$image\_save\_func = 'imagejpeg';

break;

case 'image/png':

$image\_create\_func = 'imagecreatefrompng';

$image\_save\_func = 'imagepng';

break;

default:

throw new Exception('Unknown image type.');

break;

}

$img = $image\_create\_func($originalFile);

list($width, $height) = getimagesize($originalFile);

$tmp = imagecreatetruecolor($newwidth, $newheight);

imagecopyresampled($tmp, $img, 0, 0, 0, 0, $newwidth, $newheight, $width, $height);

if (file\_exists($targetFile)) {

unlink($targetFile);

}

$image\_save\_func($tmp, "$targetFile");

}

* 1. Руководство пользователя

Приведём описание для пользователя. Сайт состоит из одной страницы *index.php.* Данная страница предоставлена на рисунке 3.2.

Graphical user interface, website

Description automatically generated

Рисунок 3.2 — Страница *index.html*

На данной странице пользователь видит уменьшенные копии загруженных им изображений. Кнопка «*upload image*» не может быть нажата без загрузки файла. Так же при загрузке файла в браузере файлов будут подсвечиваться (если ОС и приложение браузера это позволяют) только файлы того расширения, которое поддерживает форма загрузки. Пример этого приведен на рисунке 3.3.

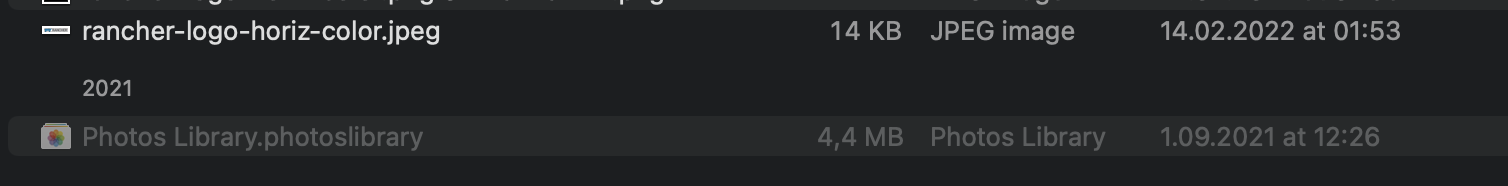


Рисунок 3.3 — Подсветка файлов поддерживаемого расширения

После успешной загрузки изображения пользователь получает уведомление об этом. Пример этого приведен на рисунке 3.4.

Text

Description automatically generated with medium confidence

Рисунок 3.4 — Уведомление об успешной загрузке

Если пользователь загрузит сломанный файл, он получит уведомление об ошибке загрузки, оно приведено на рисунке 3.5.

Text

Description automatically generated with medium confidence

Рисунок 3.5 — Уведомление о неуспешной загрузке

Если пользователь загрузит файл, размером больше десяти мегабайт, он получит уведомление об ошибке, оно приведено на рисунке 3.6.

Shape

Description automatically generated with low confidence

Рисунок 3.6 — Уведомление о неуспешной загрузке

Чтобы открыть оригинал изображения, пользователю достаточно кликнуть на картинку, и он будет перенаправлен на страницу *photo.php,* она показана на рисунке 3.7.

A screenshot of a video game

Description automatically generated with medium confidence

Рисунок 3.7 — Страница *photo.php*

Заключение

*HTTP* — лёгкий в использовании расширяемый протокол. Структура клиент-сервера, вместе со способностью к простому добавлению заголовков, позволяет *HTTP* протоколу продвигаться вместе с расширяющимися возможностями Сети.

*PHP* позволяет создавать качественные *web*-приложения за очень короткие сроки, получая продукты, легко модифицируемые и поддерживаемые в будущем.

*PHP* прост для освоения, и вместе с тем способен удовлетворить запросы профессиональных программистов.

Язык *PHP* постоянно совершенствуется, и ему наверняка обеспечено долгое доминирование в области языков *web*-программирования, по крайней мере, в ближайшее время.

В ходе выполнения лабораторной работы я изучил использование функций работы с файлами в *PHP*, разработал пример приложения с загрузкой и чтением изображений из локальной директории.