128 рецептов

php

Автор: Ипатов Евгений

```php
<?php
/**
 * Строки
 * Числа
 * Массивы
 * Дата и время
 * Работа с почтой
 * Файлы и папки
 * Графика
 * Работа с БД
 *
 * @author Ипатов Евгений
 * @version 1.0
 */
```

http://vk-book.ru

Оглавление

Введение

Это не книга по теории программирования! Это сборник рецептов, которые могут пригодиться в повседневной работе программиста. Поэтому тут нет описания способов установки и настройки рабочей среды, нет лишней теории, которой и так полно в каждой книге и на просторах интернет, нет разглагольствований по поводу красоты кода. Сборник, как можно понять из названия, содержит 128 глав(или рецептов) разбитых на восемь разделов. Каждый рецепт - это полностью рабочий код, его можно просто копировать и использовать в рабочих проектах. В каждом рецепте добавлено, минимально короткое пояснение, и максимально подробные комментарии, для, практически, каждой строки.

Сборник предназначен как для начинающих php программистов, так и для разработчиков среднего уровня. Все, что нужно знать новичкам, прежде чем начать читать и изучать рецепты – это то, как запускаются php скрипты.
Рецепты расположены в таком порядке, что новичок, читая все от начала и до конца, будет пополнять свой багаж знаний php постепенно. С каждой новой главой добавляются новые функции, и рецепты становятся сложней и сложней.
Программистам среднего уровня, которые уже многое знают из приведенных рецептов, не обязательно читать главы по порядку, достаточно, пользуясь оглавлением, найти решение на интересующую задачу.

I. Строки

1. Посчитать количество символов в строке

Для подсчета количества символов в строке существует две функции: **strlen** и **mb_strlen**. Для латиницы подойдут две функции, не зависимо от кодировки текста. А для кириллицы корректней использовать функцию **mb_strlen**. При использовании функции с префиксом **mb_**, необходимо указать кодировку текста. Если этого не сделать, то кириллические символы могут быть посчитаны не корректно.

Пример:

```
// исходная строка
$str = "Любой текст";
// определяем длину с помощью strlen
$str_len = strlen($str);
// определяем длину с помощью mb_strlen
$mb_str_len = mb_strlen($str, 'utf-8');
// вывод результата
echo "strlen: $str_len<br/>";
echo "strlen: $mb_str_len";
```

2. Удаление символов в начале и конце строки

Чтобы реализовать удаление символов в начале, в конце или и в начале и в конце строки существуют специальные функции - **ltrim**, **rtrim**, и **trim**. **Ltrim** удаляет все заданные символы в начале строки, **rtrim** в конце, а **trim** и в начале и в конце. Эти функции принимают два параметра, строку, которую необходимо

обработать и символ, который будет удален. Если символ не задан, то по умолчанию будет задан пробел.

Пример:

```
// исходная строка
$str = "111Любой текст111";
// удаляем символы в начале строки
$ltrim = ltrim($str, '1');
// удаляем символы в конце строки
$rtrim = rtrim($str, '1');
// удаляем символы в начале и конце строки
$trim = trim($str, '1');

// результат
echo "ltrim: $ltrim<br/>";
echo "rtrim: $rtrim<br/>";
echo "trim: $trim";
```

3. Удаление символов в строке

В предыдущем рецепте были рассмотрены, функции, которые удаляют символы в начале и в конце строки. Но бывает необходимо удалить символ еще и внутри самой строки. Для этого можно использовать функцию **str_replace**, она выполняет замену в строке. Но и для удаления тоже подойдет, просто заменим нужный символ на пустой.

Пример:

```
// исходная строка
$str = "Любой текст";
// символ, который будет удален
$lit = "т";
// удаляем символы
$str = str_replace($lit, "", $str);
// выводим результат
```

```
echo $str;
```

4. Получение подстроки

Для получения подстроки в php можно использовать две функции – **substr** и **mb_substr**. При работе с кириллицей лучше использовать **mb_substr**. Единственное отличие этих функций заключается, в том, что **mb_substr** принимает еще один параметр, который задает кодировку обрабатываемой строки.

Пример:

```
// исходная строка
$str = "Любой текст";
// Берем 3 символа начиная с
// третьего(отсчет идет от нуля)
$sub_1 = mb_substr($str, 2, 3, 'utf-8');
// Берем подстроку, начиная с первого символа
// и до третьего с конца
$sub_2 = mb_substr($str, 1, -3, 'utf-8');
// результат
echo $sub_1 . "<br/>";
echo $sub_2;
```

5. Разбить предложение на слова

Чтобы разбить строку на отдельные слова, можно воспользоваться функцией – **strtok**. Она принимает два параметра: обрабатываемую строку и символ, по которому будет о разделение строки на части. Разделителей может быть несколько, тогда они указываются подряд.

Пример:

```php
// исходная строка
$string = "Любой\tтекст\nдля     примера";
// массив, в котором будут записаны слова
$array_words = array();
// разбиваем строку
// разделителем выступает пробел, табуляция
// и перенос строки
$tok = strtok($string, " \t\n");
// разбиваем строку, пока не кончится предложение
while($tok) {
    $array_words[] = $tok;
    $tok = strtok(" \t\n");
}
// вывод результата
var_dump($array_words);
```

В примере был использован цикл **while**, он будет выполняться, пока условие принимает значение true, таким образом, строка будет разбиваться на слова, пока не закончатся слова.

6. Разбить строку на подстроки

Прежде чем разбить строку на подстроки необходимо определиться, по какому критерию будет разбита строка. Это может быть разделитель, например, запятая. Также это может быть регулярное выражение, например, разбивать строку, если в тексте встречаются числа.

В случае с разделителем можно использовать функцию **explode**. А при разделении с помощью регулярных выражений можно использовать **preg_split**.

Пример:

```php
// исходная строка
$string = "Строка,разделенная,запятыми";
```

```php
// разбиваем строку по разделителю - запятой
$array_words_1 = explode(',', $string);
// вывод результата
var_dump($array_words_1);

// исходная строка
$string = "Строка1разделенная2числами3еще текст";
// разбиваем строку по регулярному выражению
// '/\d/' - является регулярным выражением
$array_words_2 = preg_split('/\d/', $string);
// вывод результата
var_dump($array_words_2);
```

7. Объединить элементы массива в строку

Результатом выполнения предыдущего рецепта был массив, в каждом элементе которого хранятся отдельные части строки. Не редко бывает необходимо, наоборот, собрать все элементы массива в строку. Для этих целей можно использовать функцию **implode**. Она позволяет собрать все элементы массива в строку и при необходимости задает разделитель между элементами.

Пример:

```php
// исходный массив
$array = array(
   'Текст',
   'который',
   'разбит',
   'на',
   'элементы',
   'массива'
);
// собираем элементы массива в строку
// и разделяем каждый элемент пробелом
$string = implode(' ', $array);
// вывод результата
echo $string;
```

8. Управление регистром

Для смены регистра в строке существует не малое количество функций.
Чтобы перевести все символы строки в нижний регистр существует две функции – **strtolower** и **mb_strtolower**. При работе с кириллическими строками в кодировке utf-8, стоит использовать функцию **mb_strtolower**, а для остальных случаев можно использовать просто **strtolower**.

Пример:

```
// исходная строка
$string = "строка из ЧЕТЫРЕХ слов";
// перевод всех символов в нижний регистр
$mb_lower = mb_strtolower($string, "utf-8");
$lower = strtolower($string);
echo "mb_lower: $mb_lower<br/>";
echo "lower: $lower<br/>";
```

Для перевода всех символов строки в верхний регистр, так же существует две функции – **strtoupper** и **mb_strtoupper**. При работе с кириллическими строками в кодировке utf-8, стоит использовать функцию **mb_strtoupper**, а для остальных случаев можно использовать просто **strtoupper**.

Пример:

```
// исходная строка
$string = "строка из ЧЕТЫРЕХ слов";
// перевод всех символов в нижний регистр
$mb_upper = mb_strtoupper($string, "utf-8");
$upper = strtoupper($string);
echo "mb_upper: $mb_upper<br/>";
echo "upper: $upper<br/>";
```

Существует еще несколько функций для перевода символов в верхний регистр: **ucfirst** – преобразовывает только первый символ в строке. И **ucwords**, которая переводит в верхний регистр первый символ каждого слова строки. Но, как и с другими функциями, работающими со строками, есть проблема при работе с кириллицей в кодировке utf-8. Поэтому, стоит использовать эти функции только для обработки латиницы. А при работе с кириллицей корректней будет использовать функцию **mb_convert_case**. Она может выполнять сразу несколько операций: переводить всю строку в верхний или нижний регистр и переводить в верхний регистр все первые символы каждого слова в строке.

Пример:

```
// исходная строка
$string = "text latin";

// перевод в верхний регистр первого символа строки
$ucfirst = ucfirst($string);
// вывод результата
echo "ucfirst: $ucfirst<br/>";

// перевод первых символов каждого слова
// в верхний регистр
$ucwords = ucwords($string);
// вывод результата
echo "ucwords: $ucwords<br/>";


// исходная строка
$string = "строка из ЧЕТЫРЕХ слов";

// перевод всех символов в верхний регистр.
// Кодировка UTF-8
$upper = mb_convert_case($string, MB_CASE_UPPER, "UTF-
8");
// вывод результата
echo "upper: $upper<br/>";
```

```php
// перевод всех символов в нижний регистр.
// Кодировка UTF-8
$lower = mb_convert_case($string, MB_CASE_LOWER ,
"UTF-8");
// вывод результата
echo "lower: $lower<br/>";

// перевод первых символов каждого слова
// в верхний регистр. Кодировка UTF-8
$case_title = mb_convert_case($string, MB_CASE_TITLE ,
"UTF-8");
// вывод результата
echo "case_title: $case_title<br/>";

// перевод в верхний регистр первого символа строки
// получаем первый символ строки
$first_char = mb_substr($string, 0, 1, 'UTF-8');
// переводим первый символ в верхний регистр
$first_upper = mb_convert_case($first_char, 'UTF-8');
// берем от строки все символы, кроме первого
$all_characters = mb_substr($string, 1,
mb_strlen($string), 'UTF-8');
// соединяем первый символ и все остальные
$result = $first_upper . $all_characters;
// вывод результата
echo "result: $result<br/>";
```

9. Вывод строки в обратном порядке

Для вывода строки в обратном порядке существует функция **strrev**. Но, как и многие другие функции не корректно обрабатывает кириллицу. К сожалению, модифицированной функции для работы с любой кодировкой нет. Поэтому приходится ее дописывать самим.

Пример:

```php
// исходная строка
```

```php
$string = "text latin";
// переворачиваем строку задом на перед
$strrev = strrev($string);
// вывод результата
echo "strrev: $string<br/>";

// исходная строка
$string = "исходная строка";
// переворачиваем строку в кодировке UTF-8
// задом на перед
$mb_strrev = "";
for($i = mb_strlen($string, "UTF-8"); $i >= 0; $i--){
  $mb_strrev .= mb_substr($string, $i, 1, "UTF-8");
}
// вывод результата
echo "mb_strrev: $mb_strrev";
```

10. Определение количества подстрок

Чтобы получить количество подстрок, входящих в строку можно воспользоваться двумя функциями, они обе работают корректно: **substr_count** и **mb_substr_count**.

Пример:

```php
// исходная строка
$string = "исходная строка для примера подсчета
подстрок";
// искомая подстрока
$substr = "стр";
// подсчет вхождений
$count = substr_count($string, $substr);
$mb_count = mb_substr_count($string, $substr, "UTF-
8");
echo "count: $count<br/>";
echo "mb_count: $mb_count<br/>";
```

11. Поиск позиции первого вхождения подстроки

Поиск позиции первого вхождения подстроки можно реализовать, воспользовавшись одной из двух php функций: **strpos** или **mb_strpos**. Первая функция некорректно работает при поиске вхождения в кириллической строке в кодировке utf-8. Вторая функция работает со строками в любой кодировке, только ей необходимо передавать параметр, в котором указана кодировка обрабатываемой строки.

Пример:

```
// исходная строка
$string = "исходная строка для примера подсчета
подстрок";
// искомая подстрока
$substr = "стр";
// определение позиции первого вхождений
$strpos = strpos($string, $substr);
// определение позиции первого вхождений с учетом
кодировки
$mb_strpos = mb_strpos($string, $substr, 0, "UTF-8");
// вывод результата
echo "strpos: $strpos<br/>";
echo "mb_strpos: $mb_strpos<br/>";
```

12. Сокращение длинной строки

При выводе строк не редко бывает необходимо вместить текст в определенный размер. Но строки могут иметь различную длину, в том числе и слишком большую. В таких случаях необходимо укоротить строку. А для более читабельного вида добавить в конце многоточие. Для реализации необходимо получить длину строки,

сравнить ее размер с допустимой длиной и при необходимости обрезать и приписать в конце многоточие.

Пример:

```
// тестовая строка
$str = "Строка с очень длинным текстом, который
необходимо укоротить!";
// предел вывода текста
$count_max = 15;
// в эту переменную запишем результат
$result_str = "";
// если строка, больше предела + 3(3 точки в конце)
if(mb_strlen($str, 'utf-8') > ($count_max + 3)){
    // обрезаем строку
    $sub_str = mb_substr($str, 0, $count_max,
    'utf-8');
    // удаляем пробелы в начале и конце
    // обрезанной строки, и дописываем
    // в конец три точки
    $result_str = trim($sub_str) . "...";
    // добавляем к тексту атрибут title,
    //для вывода подсказки при наведении
    $result_str = "
        <span title='$str'>$result_str</span>
    ";
}else{
    // если строка и так короткая
    $result_str = $str;
}

// вывод результата
echo $result_str;
```

II. Числа

1. Определение максимального и минимального числа

Для определения максимального и минимального числа в php есть две удобные функции: **min** и **max**. Они могут принимать любое количество чисел и возвращать минимальное или максимально из них.

Пример:

```
// Пример 1, сравнить два числа
$a = 1;
$b = 3;
$min = min($a, $b);
$max = max($a, $b);
echo "min: $min<br/>";
echo "max: $max<br/>";

// Пример 2, сравнить 7 чисел
$min = min(1, 2, -5, 6, 0, -8, 7);
$max = max(1, 2, -5, 6, 0, -8, 7);
echo "min: $min<br/>";
echo "max: $max";
```

2. Целочисленное деление с остатком

Чтобы получить целую часть от деления можно воспользоваться функцией **intval**, которая возвращает целое значение переменной. А для определения остатка от деления используется арифметический оператор - %, например $a % $b.

Пример:

```
$a = 5;
$b = 2;
// получение целой части от деления
$intval = intval($a / $b);
```

```
// получение остатка от деления
$modulo = $a % $b;
echo "intval :$intval<br/>";
echo "modulo :$modulo";
```

3. Проверка числа на четность и нечетность

Чтобы проверить является число четным или не четным, достаточно разделить его на 2 и проверить остаток. Если остаток равен нулю, значит число четное, в противном случае число является не четным.

Пример:

```
$a = 5;
if(($a % 2) == 0){
   echo "Число четное";
}else{
   echo "Число не четное";
}
```

4. Вывод числа кратного N

Для определения числа кратного N, можно воспользоваться возможностью получения остатка от деления. Если при делении числа на N, остаток равен нулю, значит, число является кратным. Только необходимо добавить еще одно условие – проверка числа на ноль, поскольку 0/N = 0, без остатка.

Пример:

```
$n = 3;
for($i = 0; $i < 10; $i++){
   if((($i % $n) == 0) AND ($i != 0)){
         echo $i . "<br/>";
   }
```

```
}
```

5. Перевод чисел в разные системы исчисления

Для осуществления перевода между самыми популярными системами счисления в php существую несколько функций, они принимают значение в одной системе и возвращают результат уже в другой.

Пример:

```
$dec = 123;
echo "dec: $dec<br/>";

// перевод из десятичной системы счисления в двоичную
$bin = decbin($dec);
echo "bin: $bin<br/>";

// перевод из двоичной системы счисления в десятичную
$dec = bindec($bin);
echo "dec: $dec<br/>";

// перевод из десятичной системы счисления
// в восьмеричную
$oct = decoct($dec);
echo "oct: $oct<br/>";

// перевод из восьмеричной системы счисления в
десятичную
$dec = octdec($oct);
echo "dec: $dec<br/>";

// перевод из десятичной системы счисления в
шестнадцатеричною
$hex = dechex($dec);
echo "hex: $hex<br/>";

// перевод из шестнадцатеричной системы счисления в
десятичную
```

```
$dec = hexdec($hex);
echo "dec: $dec<br/>";
```

6. Округление чисел с плавающей точкой до целых

Чтобы округлить число в php существует несколько функций: **round**, **ceil** и **floor**. Функция **round** округляет число в большую или меньшую сторону в зависимости от значения дробной части, если больше или равно пяти, то округление осуществляется в большую сторону, иначе в меньшую. Функция **ceil**, независимо от дробной части, округляет число в большую сторону. **Floor** также, как и **ceil** не обращает внимание на дробную часть, только округляет число в меньшую сторону.

Пример:

```
$number = 1.2345;
$number_2 = 1.5678;

// округление числа
$round = round($number);
echo "round: $round<br/>";
$round_2 = round($number_2);
echo "round_2: $round_2<br/>";

// округление числа в меньшую сторону
$ceil = ceil($number);
echo "ceil: $ceil<br/>";

// округление числа в большую сторону
$floor = floor($number);
echo "floor: $floor<br/>";
```

7. Округление чисел с плавающей точкой до десятых и сотых

Для округления чисел, до определенного количество знаков после запятой, в php можно воспользоваться, ранее описанной, функцией **round**. Помимо параметра, в котором передается число, можно передать еще один параметр, в котором указывается, сколько знаком после запятой необходимо оставить.

Пример:

```
$number = 1.2345;
$number_2 = 1.5678;

// округление числа до десятых
$round = round($number, 1);
echo "round: $round<br/>";

// округление числа до сотых
$round = round($number_2, 2);
echo "round: $round<br/>";
```

8. Вывод слов в разной форме(1 день, 2 дня, 5 дней)

Не редко возникает необходимость вывода числа с текстовой подписью. Например, вывод количества дней - 1 день, 2 дня, 5 дней. Как можно заметить, форма слова меняется в зависимости от числа. Чтобы не писать условия для каждого случая, можно составить универсальное условие.

Пример:

```
// функция
```

```php
function numForm($number, $forma1, $forma2, $forma3){
    if(($number == "0")
         or (($number >= "5")
         and ($number <= "20"))
         or preg_match("|[056789]$|",$number)
       ){
         return "$number $forma3";
    }

    if(preg_match("|[1]$|",$number)){
         return "$number $forma1";
    }

    if(preg_match("|[234]$|",$number)){
         return "$number $forma2";
    }
}

// пример использования
$array = array(1, 2, 5);
foreach($array as $num){
  echo numForm($num, "день", "дня", "дней") . "<br/>";
}
```

В примере была использована не описанная ранее функция **preg_match**. Она позволяет осуществлять проверку значения на соответствию регулярному выражению.

9. Генерация случайных чисел

Для генерации случайных чисел в php существует две функции – **rand** и **mt_rand**, которые возвращают случайные числа в заданном диапазоне. Различия функций заключаются в методе генерации числа. **Rand** для генерации использует библиотеку операционной системы, а **mt_rand** – использует генератор самого php.

Пример:

```php
// минимальное число
$min = -100;
// максимальное число
$max = 100;
$rand = rand($min, $max);
$mt_rand = mt_rand($min, $max);

echo "rand: $rand<br/>";
echo "mt_rand: $mt_rand";
```

10. Генерация уникальных случайных чисел

В предыдущем рецепте был описан процесс генерации случайных чисел. Но бывает необходимо получить несколько случайных чисел. И не просто несколько чисел, а таким образом, чтобы все они были различными. Для реализации такого рецепта, в php нет готовых решений, поэтому необходимо сделать это самим. В основе реализации будут лежать все те же функции php, которые генерируют случайное число, только после каждой генерации необходимо выполнять проверку, существует такое число в списке или нет. Если такое число уже сгенерировано, то необходимо повторить генерацию.

Пример:

```php
// хранилище для чисел
$outArray = array();
// максимальное число
$max = 10;
// минимальное число
$min = 0;
// необходимое количество чисел
$count = 10;
// счетчик полученных чисел
$i = 0;
// цикл, который будет выполняться,
// пока не будут получены все уникальные числа
```

```php
while($i<$count){
   // генерируем случайное число
     $chislo = mt_rand($min, $max);
   // проверяем уникальность числа
     if(!in_array($chislo, $outArray)){
          // если уникальное, то записываем его в массив
          $outArray[$i] = $chislo;
          $i++;
     }
}

// вывод результата
var_dump($outArray);
```

В примере была использована ранее не описанная функция **in_array**, она позволяет осуществлять проверку наличия элемента в массиве.

11. Получение числа π (Пи)

При решении некоторых задач может потребоваться получить число Пи. Для получения этого числа в php существует два способа – вызвать функцию **pi** или получить значение предопределенной константы **M_PI**.

Пример:

```php
// с помощью функции
echo pi() . "<br/>";

// константа
echo M_PI;
```

III. Массивы

1. Заполнение массива элементами из заданного диапазона

Прежде, чем начать работу с массивом, его необходимо создать и заполнить. Не очень часто, но все же бывает необходимо, заполнить пустой массив элементами из определенного рада, с определенной последовательностью. Например, числами от 1 до 5 или буквами от a до f. На такой случай в php существуют функция **range**. Она принимает два обязательных и один не обязательный параметра. Обязательные, указывают диапазон, из которого будет заполнен массив, а необязательный определяет шаг между элементами последовательности.

Пример:

```php
// заполнение массива элементами от 1 до 10
$array = range(1, 10);
var_dump($array);

// заполнение массива элементами от 5 до -5 с шагом 2
$array = range(5, -5, 2);
var_dump($array);

// заполнение массива элементами от a до f
$array = range('a', 'f');
var_dump($array);

// заполнение массива элементами от z до a с шагом 3
$array = range('z', 'a', 3);
var_dump($array);
```

2. Заполнение массива случайными числами

Для заполнения массива случайными числами можно воспользоваться одой из двух функций генерации случайных чисел – **rand** или **mt_rand**. Так же необходимо определить размер будущего массива – его длину. Зная размер, можно выполнить цикл и на каждой итерации добавлять в массив по одному элементу, значение которого будет случайно сгенерировано.

Пример:

```
// создаем пустой массив
$array = array();
// количество элементов в массиве
$len = 10;
// минимальное значение элемента
$min = 0;
// максимальное значение элемента
$max = 0;
// заполняем массив
for($i = 0; $i < $len; $i ++){
   $array[] = mt_rand($min, $max);
}

// вывод массива
var_dump($array);
```

3. Обойти все элементы массива

Чтобы обойти каждый элемент массива, можно воспользоваться любым из циклов, но чаще всего используется **foreach**, для работы с массивами - это самый удобный цикл. Поскольку нет необходимости определять длину массива, для определения количества итераций, php это сделает за нас. А также **foreach** позволяет легко обращаться как к значениям массива, так и к их индексам – ключам.

Пример:

```php
// исходный массив
$array = array(10, 20, 30, 40, 50);
// осуществляем перебор всех элементов
foreach($array as $key => $value){
  // выводим индекс элемента и значение
  echo "key: $key value: $value <br/>";
}
```

4. Удаление элементов массива

Для удаления элементов массива в php существует функция **unset**, она позволяет удалить не только элемент, но и весь массив. После удаления элемента индексы массива не изменятся, то есть если удалить элемент с индексом 2, то индексы массива будут выглядеть следующим образом: 0, 1, 3, 4, 5. Чтобы переиндексировать массив, стоит воспользоваться функцией **array_values**, которая получает все значения, и записать их заново в массив.

Пример:

```php
// исходный массив
$array = array(10, 20, 30, 40, 50);
// удаляем элемент с индексом 2
unset($array[2]);
// выводим массив
print_r($array);
// переиндексируем массив
$array = array_values($array);
// еще раз выводим массив
print_r($array);
```

5. Минимальный и максимальный элемент массива

Определить максимальный и минимальный элемент массива можно с помощью php функций **min** и **max**. Эти функции могут

принимать набор чисел, которые необходимо сравнить, а также отлично справляются с поиском минимальных и минимальных значений в массиве.

Пример:

```
// исходный массив
$array = array(0, -10, 10, -20, 20);
// определяем минимальный элемент
$min = min($array);
// определяем максимальный элемент
$max = max($array);

// вывод результата
echo "min: $min<br/>max: $max";
```

6. Получение первого и последнего элементов массива

Для получения первого и последнего элементов массива в php существует уже готовое решение – две функции: **array_shift** и **array_pop**. Они возвращают значение первого и последнего элемента из массива.

Пример:

```
// исходный массив
$array = array(10, 20, 30, 40, 50);

// получаем первый элемент
$first = array_shift($array);
// получаем последний элемент
$last = array_pop($array);

// вывод результата
echo "first: $first <br/>";
echo "last: $last";
```

7. Удалить повторяющиеся элементы в массиве

Чтобы избавиться от повторов в массиве в php существуют уже готовая функция - **array_unique**. Помимо входного массива, функция может принимать еще один, но уже не обязательный, параметр, который определяет тип сравнения элементов: обычное сравнение элементов(используется по умолчанию), элементы сравниваются как числа, элементы сравниваются как строки.

Пример:

```
// исходный массив
$array = array(10, 20, 10, 20, 10, "a", "b", "a");

// удаление. Обычное сравнение элементов
$array_1 = array_unique($array);
print_r($array_1);

// удаление. Сравнение элементов как чисел
$array_2 = array_unique($array, SORT_NUMERIC);
print_r($array_2);

// удаление. Сравнение элементов как строк
$array_3 = array_unique($array, SORT_STRING);
print_r($array_3);
```

8. Поиск в массиве

Для поиска по массиву существует несколько удобных функций: **in_array**, **array_key_exists** и **array_search**. **In_array** позволяет быстро определить присутствует искомый элемент в массиве или нет, возвращает false или true. **Array_key_exists** определяет, присутствует индекс или ключ в массиве, возвращает

false или true. **Array_search** осуществляет поиск индекса или ключа в массиве по значению элемента, возвращает ключ или индекс, в случае успеха, иначе вернет false.

Пример:

```
// исходный массив
$array = array(10, 20, 30, 40, 50);

// искомое значение
$value = 30;
// поиск
$result = in_array($value, $array);
// вывод результата
if($result){
  echo "Элемент присутствует в массиве<br/>";
}else{
  echo "Элемент не найден в массиве<br/>";
}

// искомый индекс(ключ)
$index = 3;
// поиск
$result = array_key_exists($index, $array);
// вывод результата
if($result){
  echo "Индекс(ключ) присутствует в массиве<br/>";
}else{
  echo "Индекс(ключ) не найден в массиве<br/>";
}

// искомое значение
$value = 30;
// поиск
$index = array_search($value, $array);
// вывод результата
if($index){
  echo "Индекс элемента: $index<br/>";
}else{
  echo "Элемент не найден в массиве<br/>";
}
```

9. Объединение массивов

Массивы можно объединять несколькими способами, в зависимости от стоящей задачи. Можно объединить таким образом, что один массив станет продолжением другого, для этих целей подойдет функция **array_merge**. А можно объединить так, что один массив будет являться ключами, а другой массив будет использован в качестве соответствующих значений, в таком случае следует использовать функцию **array_combine**.

Пример:

```
// массив, значения которого будут использованы
// как ключи
$array_keys = array('key_0', 'key_1', 'key_2');
// массив, значения которого будут использованы
// как значения
$array_values = array('val_0', 'val_1', 'val_2');
// создаем новый массив
$array_combine = array_combine($array_keys,
$array_values);
// вывод результата
print_r($array_combine);


// первый массив
$array_1 = array(10, 20, 30);
// второй массив
$array_2 = array(40, 50, 60, 70);
// объединяем два массива
$array_merge = array_merge($array_1, $array_2);
// вывод результата
print_r($array_merge);
```

10. Сравнение массивов

В php существует несколько функций, для сравнения массивов, но чаще всего приходится использовать две. Первая функция – **array_diff**, осуществляет сравнение элементов одного массива с другим и возвращает массив элементов, которые не повторялись. Вторая функция – **array_intersect**, сравнивает массивы и возвращает в ответ массив элементов, которые имеются в обоих массивах.

Пример:

```
// первый массив
$array_1 = array(10, 20, 30, 40);
// второй массив
$array_2 = array(10, 50, 20, 60);
// ищем отличая первого массива, от второго
$array_diff = array_diff($array_1, $array_2);
// вывод результата
print_r($array_diff);

// ищем схожие элементы первого массива
// с элементами второго
$array_intersect = array_intersect($array_1,
$array_2);
// вывод результата
print_r($array_intersect);
```

11. Сортировка массивов

Для сортировки массивов как одномерных и многомерных, так и ассоциативных в php существует достаточно много функций. О самых часто используемых функциях по порядку:

Sort – позволяет сортировать одномерный массив по возрастанию элементов. Пример:

```
// исходный массив
```

```
$array = array(5, 1, 7, 2, 6, 3, 8, 4);
// сортировка по возрастанию
sort($array);
// вывод результата
print_r($array);
```

Rsort – сортирует одномерный массив по убыванию элементов. Пример:

```
// исходный массив
$array = array(5, 1, 7, 2, 6, 3, 8, 4);
// сортировка по убыванию
rsort($array);
// вывод результата
print_r($array);
```

Shuffle – перемешивает элементы одномерного массива в случайном порядке. Пример:

```
// исходный массив
$array = array(5, 1, 7, 2, 6, 3, 8, 4);
// перемешивает элементы в случайном порядке
shuffle($array);
// вывод результата
print_r($array);
```

Array_reverse – перемешивает элементы любого массива в обратном порядке. Если массив многомерный или ассоциативный, то значение индексов или ключей перемешиваются вместе со значениями. Пример:

```
// исходный массив
$array = array('a'=>5, 'b'=>1, 7, 'c'=>3, 4);
// сортировка в обратном порядке
$array = array_reverse($array);
// вывод результата
print_r($array);
```

Asort – сортирует любой массив по возрастанию значений. Если массив многомерный или ассоциативный, то индексы или ключи перемешиваются вместе со значениями. Пример:

```php
// исходный массив
$array = array('b' => 1, 'a' => 3, 2, 'c' => 4);
// сортировка ассоциативного массива по возрастанию
// перемешивает и значения вместе с ключами(индексами)
asort($array);
// вывод результата
print_r($array);
```

Arsort – сортирует любой массив по убыванию значений. Если массив многомерный или ассоциативный, то индексы или ключи перемешиваются вместе со значениями. Пример:

```php
// исходный массив
$array = array('b' => 1, 'a' => 3, 2, 'c' => 4);
// сортировка ассоциативного массива по убыванию
// перемешивает и значения вместе с ключами(индексами)
arsort($array);
// вывод результата
print_r($array);
```

Ksort – сортирует массив по возрастанию ключей. Значения перемешиваются вместе с ключами. Пример:

```php
$array = array('b' => 1, 'a' => 3, 'c' => 2);
// сортировка ассоциативного массива
// по возрастанию ключей(индексов)
// перемешивает и ключи(индексы) и значения
ksort($array);
// вывод результата
print_r($array);
```

12. Получение ключей и значений ассоциативного массива

Чтобы получить все ключи и значения ассоциативного массива можно воспользоваться двумя php функциями **array_keys** и

array_values. Они возвращают массивы, содержащие все ключи и значения массива.

Пример:

```
// исходный массив
$array = array(
   "key_1" => "val_1",
   "key_2" => "val_2",
   "key_3" => "val_3",
   "key_4" => "val_4",
   "key_5" => "val_5"
);

// получение всех ключей массива
$keys = array_keys($array);
// вывод результата
print_r($keys);

// получение всех значений массива
$values = array_values($array);
// вывод результата
print_r($values);
```

IV. Дата и время

1. Получить текущую дату и время

Для получения текущей даты и времени в php существует функция **date**. С ее помощью можно получать системную дату и время в огромном количестве различных форматов(год, месяц, день, час, минуты, секунды, день недели, номер дня в году, високосный год или нет и тд.). Подробнее о возможностях этой функции можно

узнать на официальном сайте php:
http://php.net/manual/ru/function.date.php

В этом же рецепте приведен код вывода самых распространенных значений даты и времени.

Пример:

```
// вывод текущего года, месяца и дня
echo date("Y-m-d");

// вывод текущего часа, минусы и секунды
echo date("H:i:s");

// вывод дня недели
echo date("l");

// номер дня недели
echo date("w");

// номер дня в году
echo date("z");

// определение високосного года
if(date("L") == 1){
   echo "Год високосный";
}else{
   echo "Год не високосный";
}
```

2. Получить текущую дату и время по Гринвичу

Чтобы получить дату и время по Гринвичу можно воспользоваться функцией **gmdate**. Эта функция идентична функции **date**, которая была описана в предыдущем рецепте. Функция может принимать все те же модификаторы, что и **date**. Единственное отличие это то, что дата и время будет формироваться по Гринвичу.

Пример:

```php
// вывод текущего года, месяца и дня
echo gmdate("Y-m-d");

// вывод текущего часа, минусы и секунды
echo gmdate("H:i:s");

// вывод дня недели
echo gmdate("l");

// номер дня недели
echo gmdate("w");

// номер дня в году
echo gmdate("z");
```

3. Получить дату первого и последнего дня месяца

Чтобы получить дату первого и последнего дня текущего месяца стоит воспользоваться функцией **date** с необходимыми модификаторами. А для получения даты первых и последних дней, например, прошлого или следующего месяца, использование одних модификаторов будет не достаточно. Необходимо в функцию **date** вторым параметром передать метку времени в формате timestamp.

Пример:

```php
// Получение даты первого дня текущего месяца
echo date('Y-m-01') . "<br/>";

// Получение даты последнего дня текущего месяца
echo date('Y-m-t') . "<br/>";

// Получение даты первого дня предыдущего месяца
echo date("Y-m-01", strtotime("-1 month")) . "<br/>";

// Получение даты последнего дня предыдущего месяца
echo date("Y-m-t", strtotime("-1 month")) . "<br/>";

// Получение даты первого дня следующего месяца
```

```
echo date("Y-m-01", strtotime("+1 month")) . "<br/>";

// Получение даты последнего дня следующего месяца
echo date("Y-m-t", strtotime("+1 month"));
```

В примере была использована, ранее не описанная функция **strtotime**, она преобразовывает текстовое значение даты(на английском языке) в метку времени timestamp.

4. Получить дату ближайшего понедельника

Получить дату ближайшего прошедшего и следующего понедельника можно, воспользовавшись теми же функциями, что и в предыдущем рецепте **date** и **strtotime**.

Пример:

```
// Дата ближайшего прошедшего понедельника
echo date("Y-m-d", strtotime("last monday"));

// Дата ближайшего следующего понедельника
echo date("Y-m-d", strtotime("next monday"));
```

5. Сравнение дат

Для сравнения двух дат, их сначала необходимо преобразовать к виду timestamp, а потом просто сравнить два полученных числа. Для преобразования даты в формат timestamp, можно воспользоваться php функцией **strtotime**.

Пример:

```
// текущая дата на сервере
```

```php
$date_1 = date("Y-m-d");
// вторая дата, с которой будет сравнение
$date_2 = "2014-10-21";

// перевод дат в формат timestamp
$date_timestamp_1 = strtotime($date_1);
$date_timestamp_2 = strtotime($date_2);

// сравниваем
if($date_timestamp_1 > $date_timestamp_2){
  echo "Первая дата больше";
}else if($date_timestamp_1 < $date_timestamp_2){
  echo "Вторая дата больше";
}else{
  echo "Даты равны";
}
```

6. Разница между датами в днях

Чтобы узнать разницу между двумя датами, необходимо обе даты перевести в вид timestamp, после чего узнать разницу. Полученное число, будет являться разницей между датами в секундах. Для перевода секунд в дни достаточно просто разделить результат на количество секунд в сутках(86400 сек).

Пример:

```php
// первая дата(текущая)
$date_1 = date("Y-m-d");
// вторая дата
$date_2 = "2014-10-31";

// перевод дат в формат timestamp
$date_timestamp_1 = strtotime($date_1);
$date_timestamp_2 = strtotime($date_2);

// разница в секундах
$diff = $date_timestamp_1 - $date_timestamp_2;
// берем модуль, возможно значение с минусом
$diff = abs($diff);
```

```
// Высчитываем количество дней
// 3600 сек = 1 час
// и округляем до целых
$diff_day = intval($diff / (3600 * 24));
// вывод количества дней
echo $diff_day;
```

В рецепте были использованы две новые функции: **abs** - получение модуля числа, **intval** - округление числа до целых.

7. Текущий день недели по-русски

В php, функция **date**, умеет выводить дату и время в огромном количестве форматов, но, к сожалению, вывод по-русски не предусмотрен. Поэтому для вывода дня недели по-русски, необходимо написать свою функцию.

Пример:

```
// функция для определения дня недели по-русски
function getDayRus(){
  // массив с названиями дней недели
    $days = array(
        'Воскресенье', 'Понедельник',
        'Вторник', 'Среда',
        'Четверг', 'Пятница', 'Суббота'
    );
  // номер дня недели
  // с 0 до 6, 0 - воскресенье, 6 - суббота
  $num_day = (date('w'));
  // получаем название дня из массива
  $name_day = $days[$num_day];
  // вернем название дня
    return $name_day;
}

// пример использования
echo getDayRus();
```

8. Название месяца по-русски

Как уже было сказано в предыдущем рецепте, php функция **date**, не умеет отдавать дату по-русски, поэтому для вывода названия месяца по-русски необходимо написать свою функцию.

Пример:

```
/*
функция для получения названия месяца по-русски
$num_month - номер месяца,
не обязательный параметр, если параметр не задан,
то функция вернет название текущего месяца
*/
function getMonthRus($num_month = false){
  // если не задан номер месяца
  if(!$num_month){
        // номер текущего месяца
        $num_month = date('n');
  }
    // массив с названиями месяцев
    $monthes = array(
        1 => 'Январь', 2 => 'Февраль', 3 => 'Март',
        4 => 'Апрель', 5 => 'Май', 6 => 'Июнь',
        7 => 'Июль', 8 => 'Август',9 => 'Сентябрь',
        10 => 'Октябрь', 11 => 'Ноябрь',
        12 => 'Декабрь'
  );
  // получаем название месяца из массива
  $name_month = $monthes[$num_month];
  // вернем название месяца
    return $name_month;
}

// пример использования
echo getMonthRus(1) . "<br/>";
echo getMonthRus();
```

9. Время выполнения скрипта

Чтобы измерить время выполнения участка кода, необходимо перед началом выполнения кода, получить текущее время, а также по завершению выполнения кода, еще раз получить время. Далее просто высчитывается разница во времени. Поскольку скорость скрипта может измеряться в тысячных и десятитысячных долях секунды, необходимо воспользоваться php функцией **microtime**. Она позволяет получать метку времени в микросекундах.

Пример:

```
// засекаем начало выполнения скрипта
$start_time = microtime(true);

// код, время которого нужно замерить
// пример скрипта
for($i = 0; $i<1; $i+=0.000001);

// засекаем завершение выполнения скрипта
$finish_time = microtime(true);

// высчитываем разницу во времени
$result_time = $finish_time - $start_time;

// форматированный вывод результата
printf('Затрачено %.4F сек.', $result_time);
```

В примере была использована, ранее не описанная функция - **printf**, она позволяет выводить отформатированную строку.

10. Определить возраст по дате рождения

Для определения возраста достаточно проверить, было уже день рождение в текущем году или нет. Если уже прошло, то необходимо из текущего года вычесть год рождения. А если день рождения еще

не прошел, то необходимо из текущего года вычесть год рождения и еще минус один год.

Пример:

```
/*
* Получение возраста
* $day - день
* $mouth - месяц
* $year - год
*/
function getAge($day, $mouth, $year){
   // если в этом году уже был день рождения
   if(    $mouth > date('m') ||
          $mouth == date('m') &&
          $day > date('d')
   ){
          return (date('Y') - $year - 1);
   }else{
          // если еще не прошел день рождения
          $result = date('Y') - $year;
   }
   return $result;
}

// пример использования
echo getAge(24, 1, 1989);
```

11. Получить знак зодиака по дате рождения

Чтобы получить знак зодиака, достаточно знать только день и месяц рождения. Алгоритм вычисления знака прост: необходимо создать массив, в котором будет храниться число месяца, в которое происходит смена знака зодиака и массив названий знаков в таком порядке, как они и начинаются с начала года. Когда массивы будут созданы, будет достаточно сравнить день рождения с днем, при котором происходит смена знака зодиака, если смена еще не

произошла, то вернем название знака, соответствующей месяцу рождения. Иначе, вернем следующий по порядку знак зодиака.

Пример:

```
/*
* определяет по дате рождения знак зодиака
* $month - месяц
* $day - день рождения
*/
function getZodiac($month, $day){
   // массив с названиями знаков зодиака
   $zodiacName = array(
         "Козерог", "Водолей", "Рыбы",
         "Овен", "Телец", "Близнецы",
         "Рак", "Лев", "Девы",
         "Весы", "Скорпион", "Стрелец"
   );
   // массив дней, с которых сменяется знак зодиака
   $zodiacDate = array(
         21, 20, 20, 20, 20, 20,
         21, 22, 23, 23, 23, 23
   );
   // если в выбранный день месяца знак уже сменился
   echo $index_m;
   if ($day < $zodiacDate[$month - 1]){
         $result = $zodiacName[$month - 1];
   }else{
         if($month == 12) $month = 0;
         $result = $zodiacName[$month];
   }
   return $result;
}

// пример использования
echo getZodiac(1, 24);
```

V. Работа с почтой

1. Отправка письма

Для отправки писем в php можно воспользоваться функцией **mail**. Эта функция принимает три обязательных параметра: e-mail получателя, тема и текст письма. Помимо обязательных параметров, есть возможность передачи еще двух не обязательных. Первый - это строка, которая будет добавлена в отправляемые заголовки письма. Во второй необязательный параметр можно передавать флаги в виде аргументов командной строки для программы, которая осуществляет отправку писем.

Пример:

```
// получатель письма
$strTo = 'test1@test.com';
// Тема письма
$subject = "Тестовое письмо";
// Текст письма.
// Тут может быть как просто текст, так и html код
$message = '
  <html>
        <head>
              <title>Тестовое письмо</title>
        </head>
        <body>
              <p>Текст письма</p>
        </body>
  </html>
';
// заголовок письма
$headers= "MIME-Version: 1.0\r\n";
// кодировка письма
$headers .= "
  Content-type: text/html; charset=utf-8\r\n
";
// от кого письмо
```

```
$headers .= "From: Тестовое письмо <no-
reply@test.com>\r\n";
// отправляем письмо
$result = mail($strTo, $subject, $message, $headers);
// результат отправки письма
if($result){
  echo "Письмо успешно отправлено";
}else{
  echo "Письмо не отправлено";
}
```

2. Отправка письма нескольким получателям

Чтобы отправить письмо сразу нескольким получателем, как и в предыдущем рецепте, можно воспользоваться php функцией **mail**. Единственное отличие от прошлого примера, будет заключаться в том, что параметр, передающий e-mail получателя, будет содержать не один адрес, а несколько, разделенных запятыми.

Пример:

```
// массив получателей письма
$arrayTo = array(
  'test1@test.com',
  'test2@test.com',
  'test3@test.com'
);
// переводим массив в строку
// и разделяем адреса запятыми
$strTo = implode(",", $arrayTo);

// Тема письма
$subject = "Тестовое письмо";
// Текст письма.
// Тут может быть как просто текст, так и html код
$message = '
  <html>
      <head>
          <title>Тестовое письмо</title>
```

```
        </head>
        <body>
            <p>Текст письма</p>
        </body>
    </html>
';
// заголовок письма
$headers= "MIME-Version: 1.0\r\n";
// кодировка письма
$headers .= "
    Content-type: text/html; charset=utf-8\r\n
";
// от кого письмо
$headers .= "From: Тестовое письмо <no-
reply@test.com>\r\n";
// отправляем письмо
$result = mail($strTo, $subject, $message, $headers);
// результат отправки письма
if($result){
    echo "Письмо успешно отправлено";
}else{
    echo "Письмо не отправлено";
}
```

3. Проверка корректности e-mail адреса

Для проверки корректности e-mail адреса в php нет стандартных функций, но осуществить проверку можно и другим способом. Самый простой – это составить регулярное выражение, которое и будет проверять e-mail.

Прежде чем, писать регулярное выражение, стоит определиться, какие адреса являются корректными. Каждый e-mail должен состоять из двух частей – имени пользователя и доменного имени, а роль разделителя играет @. Второй признак корректности — это то, что символы, составляющие имя пользователя и доменное имя, должны содержать только большие и маленькие латинские

символы, цифры, тире, нижнее подчеркивание и точки. Третий признак: обязательное наличие домена первого уровня, это ru, com, net и тд.

Определившись с тем, какой шаблон должен быть у e-mail адреса, не сложно составить регулярное выражение:

```
// e-mail адрес, который будем проверять
$email = "admin@test_site.com";

// Проверка e-mail адреса
if(preg_match("
   |^[-0-9a-z_\.]+@[-0-9a-z_^\.]+\.[a-z]{2,6}$|i"
   , $email
)){
  echo "e-mail корректный";
}else{
  echo "e-mail не корректный";
}
```

В коде была использована функция, не описанная ранее - **preg_match**. Она выполняет проверку на соответствие регулярному выражению.

4. Как определить, читали письмо или нет

Чтобы определить, читали отправленное письмо или нет, можно воспользоваться небольшой хитростью – отправить в письме картинку, которая будут подгружаться с удаленного сервера, и при обращении к этой картинке, можно реализовать вызов счетчика количества просмотров.

Для того чтобы при обращении к картинке на сервере выполнялся скрипт можно воспользоваться apache модулем mod_rewrite. Он

позволит реализовать редирект с картинки на скрипт. Для этого достаточно добавить в корень сайта файл .htaccess с кодом:

```
<IfModule mod_rewrite.c>
  RewriteEngine on
  RewriteRule ([[:alnum:]]+).png$ img.php?em=$1 [L]
</IfModule>
```

Таким образом, при обращении к картинке с адресом:

```
<img src="http://site.ru/dGVzdEBtYWlsLnJ1.png"/>
```

будет вызван скрипт http://site.ru/img.php?em=dGVzdEBtYWlsLnJ1

Как вы уже наверняка обратили внимание, помимо вызова самого скрипта, происходит передача GET параметра em – имя картинки. Это сделано специально, с помощью этого параметра можно передавать любую необходимую информацию, например e-mail адрес получателя письма. В данном примере это закодированный, с помощью base64, e-mail получателя.

Остается только написать скрипт, который будет обрабатывать обращения и возвращать в ответ картинку и записывать статистику обращений:

```
// генерируем картинку и отдаем ее
// создаем холст 1 на 1 пиксель
$image = imagecreatetruecolor(1,1);
// делаем его белым
imagefill($image, 0, 0, 0xFFFFFF);
// задаем заголовок для вывода картинки
header('Content-type: image/png');
// выводим картинку
imagepng($image);
// очищаем память от картинки
imagedestroy($image);

// проверяем наличие GET параметра
if(isset($_GET['em'])){
```

```
    // получили email пользователя,
    // который открыл письмо
    // раскодирем данные
      $email = base64_decode($_GET['em']);

    // тут реализуем запись статистики
    // в файл или базу данных
}
```

В рецепте был использован код для генерации картинки, об этом подробнее написано в V части сборника рецептов(«Графика»). Также была использована ранее не описанная функция **base64_decode**, она осуществляет декодирование данных закодированных с помощью функции **base64_encode** в формат base64.

5. Отправка писем с вложенными файлами

Для отправки писем с вложениями достаточно использовать функцию php **mail**. Файлы, которые будут отправлены, необходимо закодировать в формат base64 и добавить в тело письма, а также указать в отправляемых заголовках письма информацию о том, что в письме присутствуют файлы.

Чтобы отделить закодированный файл от текста письма, необходимо добавить текстовый разделитель, это может быть любая уникальная строка. Разделитель следует обозначить в отправляемых заголовках, и выводить до и после прикрепления файла в тексте письма.

Пример:

```
// файл
$file = "/files/file.txt";
// кому письмо
```

```
$mailTo = "zhenikipatov@yandex.ru";
// от кого письмо
$from = "test@files.com";
// тема письма
$subject = "Test file";
// текст письма
$message = "Тестовое письмо с вложением";

// разделитель в письме
$separator = "---";
// Заголовки для письма
$headers = "MIME-Version: 1.0\r\n";
// задаем от кого письмо
$headers .= "From: $from\nReply-To: $from\n";
// в заголовке указываем разделитель
$headers .= "Content-Type: multipart/mixed;" .
        "boundary=\"$separator\"";

// начало тела письма, выводим разделитель
$bodyMail = "--$separator\n";
// кодировка письма
$bodyMail .= "Content-type: text/html;" .
          "charset='utf-8'\n";
// задаем конвертацию письма
$bodyMail .= "Content-Transfer-Encoding: quoted-
printable";
// задаем название файла
$bodyMail .= "Content-Disposition: attachment;"
          . "filename==?utf-8?B?" .
           base64_encode(basename($file))."?=\n\n";
$bodyMail .= $message."\n"; // добавляем текст письма
$bodyMail .= "--$separator\n";

// тип контента и имя файла
$bodyMail .= "Content-Type: application/octet-stream;"
              . "name==?utf-8?B?" .
              base64_encode(basename($file))."?=\n";
// кодировка файла
$bodyMail .= "Content-Transfer-Encoding: base64\n";
$bodyMail .= "Content-Disposition: attachment;" .
         "filename==?utf-8?B?" .
         base64_encode(basename($file))."?=\n\n";
```

```
// считываем файл
$contentFile = file_get_contents($file);
// кодируем и прикрепляем файл
$bodyMail .=
chunk_split(base64_encode($contentFile))."\n";
$bodyMail .= "--".$separator ."--\n";

// отправка письма
$result = mail($mailTo, $subject, $bodyMail,
$headers);
if($result){
  echo "Письмо успешно отправлено";
}else{
  echo "Письмо не отправлено";
}
```

В рецепте было использовано несколько ранее не описанных функций.

Base64_encode – кодирует данные в формат base64.

Chunk_split – разбивает строку на фрагменты, осуществляет перенос каждого фрагмента на новую строку, по умолчанию длина одного фрагмента 76 символов.

Basename – определяет имя файла из указанного пути.

File_get_contents – получает содержимое файла.

6. Отправка писем с картинками в тексте

Не редко необходимо отправлять письма, в которых помимо текста должна быть отправлена html-верстка с картинками. Реализовать отображение картинок можно двумя способами: прописывать для картинок полные пути – загрузка изображений будет происходить с уделенного сайта. Или отправлять картинки вместе с письмом. Второй способ работает более корректно, поскольку при загрузке изображений с удаленных сайтов некоторые почтовые программы блокируют отображение.

Для верстки письма используются все те же теги и стили, что и при обычной верстке, за исключением того, что стили должны находиться в самой верстке, а не в подключаемых файлах. Еще одно отличие – это то, что для изображений(img) в атрибутах src необходимо прописывать не путь, а CID изображения. CID — Content-ID будет указывать на картинку, которую необходимо предварительно закодировать в base64 и отправим вместе с письмом.

Как отправлять письма с вложениями, было подробнее описано в предыдущем рецепте – «Отправка письма с вложениями».

Пример:

```
// картинки
$attach = array(
    '/imgs/1.jpg',
    '/imgs/2.jpg'
);
// чтобы отображалась картинка и ее не было в аттаче
// путь к картинке задается через CID: - Content-ID
// тестовая верстка письма
$text = '
    <div style="width: 700px; margin: 0 auto;">
        <h1>тело письма с картинкой</h1>
        <h2>Блок по центру</h2>
        <p>
        <img style="float: left;" src="cid:1.jpg" />
        Какой-то текст вокруг картинки.
        Какой-то текст вокруг картинки.
        Какой-то текст вокруг картинки.
        Какой-то текст вокруг картинки.
        <br/>
        <img style="float: left;" src="cid:2.png" />
        Какой-то текст вокруг картинки.
        Какой-то текст вокруг картинки.
        Какой-то текст вокруг картинки.
        Какой-то текст вокруг картинки.
        </p>
    </div>
```

```
';

// E-mail отправителя
$from = "test@test.com";
// E-mail получателя
$to = "test_2@test.com";
// Тема письма
$subject = "Тема письма";

// Заголовки письма === >>>
$headers = "From: $from\r\n";
//$headers .= "To: $to\r\n";
$headers .= "Subject: $subject\r\n";
$headers .= "Date: " . date("r") . "\r\n";
$headers .= "X-Mailer: zm php script\r\n";
$headers .= "MIME-Version: 1.0\r\n";
$headers .="Content-Type: multipart/alternative;\r\n";
// генерируем базовый разделитель
$baseboundary = "------------" . md5(microtime());
$headers .= "  boundary=\"$baseboundary\"\r\n";
// <<< ====================

// Тело письма === >>>
$message  =  "--$baseboundary\r\n";
$message .= "Content-Type: text/plain;\r\n";
$message .= "Content-Transfer-Encoding: 7bit\r\n\r\n";
$message .= "--$baseboundary\r\n";
// генерируем разделитель для картинок
$newboundary = "------------" . md5(microtime());
$message .= "Content-Type: multipart/related;\r\n";
$message .= "  boundary=\"$newboundary\"\r\n\r\n\r\n";
$message .= "--$newboundary\r\n";
$message .= "Content-Type: text/html; ".
            "charset=utf-8\r\n";
$message .= "Content-Transfer-Encoding: 7bit\r\n\r\n";
$message .= $text . "\r\n\r\n";
// <<< ==============

// прикрепляем файлы ===>>>
foreach($attach as $filename){
    $mimeType='image/png';
    // получаем картинку
    $fileContent = file_get_contents($filename,true);
```

```
        $filename = basename($filename);
        $message.="--$newboundary\r\n";
        $message.="Content-Type: $mimeType;\r\n";
        $message.=" name=\"$filename\"\r\n";
        $message.="Content-Transfer-Encoding: base64\r\n";
        $message.="Content-ID: <$filename>\r\n";
        $message.="Content-Disposition: inline;\r\n";
        $message.=" filename=\"$filename\"\r\n\r\n";
        // кодируем картинку
        $message.=
            chunk_split(base64_encode($fileContent));
}
// <<< ====================

// заканчиваем тело письма, дописываем разделители
$message.="--$newboundary--\r\n\r\n";
$message.="--$baseboundary--\r\n";

// отправка письма
$result = mail($to, $subject, $message , $headers);
if($result){
  echo "Письмо успешно отправлено!";
}else{
  echo "Письмо не отправлено!";
}
```

7. Отправка писем через SMTP протокол

SMTP — сетевой протокол, предназначенный для передачи электронной почты в сетях TCP/IP. Для работы с почтовыми серверами через SMTP протокол, необходимо реализовывать обращение с помощью сокетов. Для открытия сокета используется php функция **fsockopen**.

После подключения к почтовому серверу, необходимо «представиться» серверу – передать ему логин и пароль пользователя, которому доступна отправка почты. После этого

осуществляется передача e-mail адресов отправителя и получателя. И в последнюю очередь передаются «тело» письма: заголовки и содержимое.

В качестве примера, приведен код, который обращается к почтовому серверу yandex.ru:

```
// имя пользователя
$smtp_username = 'username@yandex.ru';
// пароль
$smtp_password = 'password';
// адрес smtp сервера
$smtp_host = 'ssl://smtp.yandex.ru';
// порт для обращения к smtp серверу
$smtp_port = 465;

// тема письма
$subject = "Тема письма";
// текст письма
$message = "Текст письма";
// e-mail получателя письма
$mailTo = "zhenikipatov@yandex.ru";

// заголовок письма
$headers = "MIME-Version: 1.0\r\n";
// кодировка письма
$headers .= "Content-type: text/html; " .
            "charset=utf-8\r\n";
// от кого письмо
$headers .= "From: Evgeniy <admin@vk-book.ru>\r\n";

// тело письма
$contentMail = "Date: " . date("D, d M Y H:i:s")
               . " UT\r\n";
$contentMail .= 'Subject: =?UTF-8?B?' .
                    base64_encode($subject) .
"=?=\r\n";
$contentMail .= $headers . "\r\n";
$contentMail .= $message . "\r\n";

// соединение с почтовым сервером через сокет
if(!$socket = @fsockopen(
```

```
                    $smtp_host,
                    $smtp_port,
                    $errorNumber,
                    $errorDescription,
                    30)
){
  // если произошла ошибка
  die($errorNumber.".".$errorDescription);
}

// проверяем ответ сервера, если код 220
// значит все прошло успешно
if (!parseSocketAnswer($socket, "220")){
  die('Ошибка соединения');
}

// представляемся почтовому серверу,
// передаем ему адрес своего хоста
$server_name = $_SERVER["SERVER_NAME"];
fputs($socket, "HELO $server_name\r\n");
// проверяем ответ сервера, если код 250
// значит все прошло успешно
if (!parseSocketAnswer($socket, "250")){
  fclose($socket);
  die('Ошибка при приветствии');
}

// начинаем авторизацию на почтовом сервере
fputs($socket, "AUTH LOGIN\r\n");
// проверяем ответ сервера, если код 334
// значит все прошло успешно
if (!parseSocketAnswer($socket, "334")){
  fclose($socket);
  die('Ошибка авторизации');
}

// отправляем почтовому серверу логин,
// через который будем авторизовываться
fputs($socket, base64_encode($smtp_username)."\r\n");
// проверяем ответ сервера, если код 334
// значит все прошло успешно
if (!parseSocketAnswer($socket, "334")){
  fclose($socket);
```

```
  die('Ошибка авторизации');
}

// отправляем почтовому серверу пароль
fputs($socket, base64_encode($smtp_password)."\r\n");
// проверяем ответ сервера, если код 235
// значит все прошло успешно
if (!parseSocketAnswer($socket, "235")){
  fclose($socket);
  die('Ошибка авторизации');
}

// сообщаем почтовому серверу e-mail отправителя
fputs($socket, "MAIL FROM: <".$smtp_username.">\r\n");
// проверяем ответ сервера, если код 250
// значит все прошло успешно
if (!parseSocketAnswer($socket, "250")){
  fclose($socket);
  die('Ошибка установки отправителя');
}

// сообщаем почтовому серверу e-mail получателя
fputs($socket, "RCPT TO: <" . $mailTo . ">\r\n");
// проверяем ответ сервера, если код 250
// значит все прошло успешно
if (!parseSocketAnswer($socket, "250")){
  fclose($socket);
  die('Ошибка установки получателя');
}

// сообщаем почтовому серверу,
// что сейчас начнем передавать данные письма
fputs($socket, "DATA\r\n");
// проверяем ответ сервера, если код 354
// значит все прошло успешно
if (!parseSocketAnswer($socket, "354")){
  fclose($socket);
  die('Ошибка при передачи данных письма');
}

// передаем почтовому серверу данные письма
fputs($socket, $contentMail."\r\n.\r\n");
// проверяем ответ сервера, если код 250
```

```
// значит все прошло успешно
if (!parseSocketAnswer($socket, "250")){
  fclose($socket);
  die("Ошибка при передачи данных письма");
}

// сообщаем почтовому серверу,
// что закрываем соединение
fputs($socket, "QUIT\r\n");
// закрываем соединение
fclose($socket);

// результат отправки
echo "Письмо успешно отправлено";


// функция, которая будет анализировать ответ
// почтового сервера
// Ищет в ответе сервера необходимый код
function parseSocketAnswer($socket, $response) {
  while (@substr($responseServer, 3, 1) != ' ') {
        if (!($responseServer = fgets($socket, 256))){
              return false;
        }
  }
  if (!(substr($responseServer, 0, 3) == $response)) {
        return false;
  }
  return true;
}
```

8. Получить письма. Пример работы с IMAP протоколом

IMAP — протокол для доступа к электронной почте. Через этот протокол можно получать любую информацию о почте пользователя. Для работы с почтовым сервером через протокол

IMAP в php существует много функций. В примере используем только несколько основных:

imap_open – открывает соединение с почтовым сервером по протоколу IMAP.
imap_search – осуществляет поиск писем по заданным параметрам, например, «NEW» - найдет все новые. И возвращает массив номеров писем.
imap_header – возвращает заголовки письма по его номеру.
imap_fetchbody – получает содержимое «тела» письма по его номеру.
imap_close – закрывает соединение с почтовым сервером.
imap_last_error – возвращает последнюю IMAP-ошибку.

Пример:

```
// логин
$email = "username@yandex.ru";
// пароль
$password = "password";
// соединяемся с почтовым сервером,
// в случае ошибки выведем ее на экран
$connect_imap = imap_open("
   {imap.yandex.ru:993/imap/ssl}INBOX",
   $email,
   $password
) or die("Error:" . imap_last_error());
// проверим ящик на наличие новых писем
$mails = imap_search($connect_imap, 'NEW');
// если есть новые письма
if($mails){
   // перебираем все письма
   foreach($mails as $num_mail){
         // получаем заголовок
         $header = imap_header(
               $connect_imap, $num_mail
         );
         // достаем ящик отправителя письма
         $mail_from = $header->sender[0]->mailbox .
               "@" . $header->sender[0]->host;
```

```
            echo "От кого: $mail_from <br/>";
            // получаем тему письма
            $subject = $header->subject;
            echo "Тема письма: $subject <br/>";
            // получаем содержимое письма
            $text_mail = imap_fetchbody(
                  $connect_imap, $num_mail, 1
            );
            echo "Тело письма: $text_mail <br/>";
            echo "<hr/>";
    }
}else{
    echo "Нет новых писем";
}
// закрываем соединение
imap_close($connect_imap);
```

VI. Файлы и папки

1. Создание файлов

Начать изучение работы с файлами и папками, стоит с самого простого – создание. Создание файлов можно реализовать с помощью функции **fopen**.

Пример:

```
// создать файл и записать в него
// путь до файла и его название
$file_name = "/file.txt";
// открываем файл, если файл не существует,
//делается попытка создать его
$file = fopen($file_name, "w+");
// записываем в файл текст
```

```php
$text = "text";
fwrite($file, $text);
// закрываем файл
fclose($file);
```

Помимо **fopen,** в примере были использованы еще две функции - **fwrite** и **fclose**. Они позволяют сделать запись в файл и закрыть его.

2. Чтение из файлов

Для чтения из файла, понадобятся функции **feof** и **fgets**, которые определяют, прочитан ли файл до конца и считывают строку из файла. То есть, чтобы прочитать весь файл, необходимо его открыть и в цикле - **while**, который будет выполняться до тех пор, пока файл не будет прочтен до конца, считывать данные построчно.

Пример:

```php
// чтение из файла
// путь до файла и его название
$file_name = "/file.txt";
// открываем файл
$file = fopen($file_name, 'r');
$str = "";
// считываем все из файла
while (!feof($file)) {
   $buffer = fgets($file, 1024);
   $str .= $buffer;
}
fclose($file);
echo $str;
```

3. Удаление файлов

Чтобы удалить файл, необходимо воспользоваться функцией **unlink**.

Пример:

```
// удаление файла
// путь до файла и его название
$file_name = "/file.txt";
if (is_readable($file_name)){
   $result = unlink($file_name);
   if($result){
         echo "Файл удален!";
   }else{
         echo "Файл НЕ удален!";
   }
}else{
   echo 'Файл не существует или нет к нему доступа.';
}
```

В примере также была использована функция **is_readable**, она определяет доступность файла.

4. Копирование файлов

Для копирования файлов в php имеется специальная функция - **copy**. С ее помощью можно легко осуществить копирование файла.

Пример:

```
// расположение оригинала
$file = 'folder_1/1.txt';
// расположение копии
$new_file = 'folder_1/2.txt';

// создание копии файла
if (copy($file, $newfile)) {
    echo "Файл успешно скопирован!";
}else{
    echo "Файл не удалось скопировать!";
```

```
}
```

5. Переименование файлов

Чтобы переименовать файл, можно использовать php функцию - **rename**.

Пример:

```
// файл, который будет переименован
$file = 'folder_1/1.txt';
// новое имя
$new_name = 'folder_1/2.txt';
// переименовываем
if (rename($file, $new_name)) {
    echo "Файл успешно переименован!";
}else{
    echo "Файл не удалось переименовать!";
}
```

6. Перемещение файлов

Для перемещения файлов, в php нет отдельных функций, но можно воспользоваться **rename**, она прекрасно выполняет перемещение файлов.

Пример:

```
// расположение файла
$file = 'folder_1/1.txt';
// куда будет перемещен
$new_file = 'folder_2/1.txt';
if (rename($file, $new_file)) {
    echo "Файл успешно перемещен!";
}else{
    echo "Файл не удалось переместить!";
}
```

Помимо использования функции **rename**, для перемещения файлов можно придумать еще несколько способов. Например, осуществить копирование файла, а потом удалить оригинал.

Пример:

```
$file = 'folder_1/1.txt';
$new_file = 'folder_2/1.txt';
copy($file, $new_file); // делаем копию
unlink($file); // удаляем оригинал
```

7. Получение размера файла

Для получения размера файла достаточно одной функции – **filesize**. Результатом этой функции будет размер проверяемого файла в байтах.

Пример:

```
// размер файла
// путь до файла и его название
$file_name = "/file.txt";
// получение размера файла
$size = filesize($file_name);
echo $size;
```

8. Размер файла. Перевод байт в КБ, Мб и тд

В приведенном выше примере – получение размера файла, результат будет выведен в байтах, это несколько неудобно, и чтобы вывести результат в читабельном виде, можно написать небольшую функцию для форматированного вывода. Эта пользовательская функция работает очень просто – определяет количество разрядов

байт, и в зависимости от полученного результата возвращает нужное название единицы измерения информации.

Пример:

```
// размер файла
// путь до файла и его название
$file_name = "/file.txt";
$size = filesize($file_name);
// вызываем функцию для форматирования размера файла
echo format_size($size);

// функция форматирует вывод размера файла
function format_size($size){
   $metrics[0] = 'байт';
   $metrics[1] = 'Кбайт';
   $metrics[2] = 'Мбайт';
   $metrics[3] = 'Гбайт';
   $metrics[4] = 'Тбайт';
   $metric = 0;
   while(floor($size / 1024) > 0){
          $metric ++;
          $size /= 1024;
   }
   $result = round($size, 1) . " " .
    (isset($metrics[$metric]) ? $metrics[$metric] : '???');
   return $result;
}
```

9. Получить расширение (формат) файла. Способ 1

В php не редко приходится обрабатывать файлы, но чтобы корректно их читать или редактировать необходимо в первую очередь знать с каким файлом происходит работа. Для этого нужно получить расширение файла – его формат.

Пример:

```
// получение формата файла
$fileName = 'file.txt';
```

```
echo getFormat_1($fileName);

// способ 1
function getFormat_1($fileName) {
   $arr = explode(".", $fileName);
   return end($arr);
}
```

В приведенном примере, получение формата файла происходит с помощью двух функций – **explode** и **end**. Первая функция разбивает название файла на массив, разделителем в данном случае является точка, а с помощью второй функции - **end** извлекается последний элемент полученного массива.

10. Получить расширение (формат) файла. Способ 2

Для получения формата файлов можно использовать множество разных способов, и в подтверждение этому, ниже приведен еще один пример.

Пример:

```
// получение формата файла
$fileName = 'file.txt';
echo getFormat_2($fileName);

// способ 2
function getFormat_2($fileName) {
   return preg_replace('/^.*\.(.*)$/U', '$1',
$fileName);
}
```

В основе этого способа лежат регулярные выражения, используя которые, можно легко получить расширение файла.

11. Получить расширение (формат) файла. Способ 3

Как выше говорилось, способов для определения формата файлов очень много. Ниже приведен еще один способ, но это далеко не последний.

Пример:

```
// получение формата файла
$fileName = 'file.txt';
echo getFormat_3($fileName);

// способ 3
function getFormat_3($fileName) {
   $info = pathinfo($fileName);
   return $info['extension'];
}
```

Этот способ работает, используя функцию – **pathinfo**, которая возвращает информацию о пути к файлу.

12. Удаление строки из файла

При работе с текстовыми файлами, может возникнуть необходимость в удалении строк. Следующий рецепт поможет это реализовать. В примере приведена пользовательская функция, которая принимает два параметра. Первый – путь до файла, а второй – номер строки, которую необходимо удалить.

Пример:

```
// пример использования функции удаления сроки
$result = delStringByNum('/files/test.txt', 3);
var_dump($result);
```

```php
/**
* Функция для удаления строки из файла
*
* @param string $fileName - расположение файла
* @param string $num - номер строки,
* которую нужно удалить
*/
function delStringByNum($fileName, $num){
   // Открываем файл
   $file = @file($fileName);
   // если файл не найден, выводим ошибку
   if(!$file){
         trigger_error("File '$fileName' not found!");
         return false;
   }
   // если номер строки не корректный, сообщим об этом
   if(!isset($file[$num-1])){
         trigger_error("
               Incorrect number string: ($num)
         ");
         return false;
   }
   // удаляем строку
   $num = intval($num)-1;
   unset($file[$num]);
   // открываем файл для записи
   $fileOpen = @fopen($fileName,"w");
   // если файл невозможно редактировать,
   // сообщаем об этом
   if(!$file){
         trigger_error("
               File '$fileName' is not writable!
         ");
         return false;
   }
   // перезаписываем файл
   fputs($fileOpen, implode("", $file));
   fclose($fileOpen);
   return true;
}
```

В приведенном примере использовалась, ранее не описываемая, php функция - **trigger_error**, она позволяет выводить пользовательские ошибки, предупреждения и уведомления. Что очень удобно при написании пользовательских функций, которые в ходе выполнения могут вернуть ошибку, например, при обращении к несуществующему файлу.

13. Генерация уникального имени файла

При разработке часто приходится генерировать множество файлов, как временных, так и постоянных - рабочих. И не редко в связи с этим может возникнуть конфликты при создании файлов, поскольку имена могут совпадать. Чтобы решить эту проблему, можно перед созданием файла, проверять наличие такого имени. И если название уже существует, то приписывать к имени файла префикс, например, так: “123_file_name.txt”.

Пример:

```
// пример использования
print_r(getUniqName('./files/', 'test_file.txt'));
// при существовании test_file.txt
// результат будет - 1_test_file.txt

/**
* Получение уникального имени для файла
*
* @param string $path - путь к папке,
*  где будет проверка
*
* @param string $fileName - исходное имя файла
*
* @return string - уникальное имя файла
*/
function getUniqName($path, $fileName){
   $num = 1; // счетчик
   // проверяем, существует ли файл с таким именем
   if(file_exists($path . $fileName)) {
         // добавляем префикс и проверяем
```

```
        // наличие файла с таким именем
        while(file_exists($path.$num.'_'.$fileName)){
              $num ++;
        }
        return $num . '_' . $fileName;
  }else{
        return $fileName;
  }
}
```

В этом примере была использована, ранее не описанная функция - **file_exists**, она осуществляет проверку существования файла.

14. Удаление временных файлов

Практически с той же регулярностью, что и создание файлов, в процессе разработки возникает необходимость в удалении файлов. Про удаление уже был рецепт, но в нем удаление осуществлялось по одному файлу. В этом же примере, показано, как можно осуществлять массовое удаление файлов. В основном под массовое удаление попадают временные файлы, которые используются короткий промежуток времени, после чего просто занимают место на сервере.

Пример:

```
// путь до папки с временными файлами
$folderPath = "/tmp";
$count = 0; // счетчик файлов
// проверяем существование
if (is_dir($folderPath)) {
  // открываем папку
  if ($dir = opendir($folderPath)) {
        // перебираем все файлы
        while (($file = readdir($dir)) !== false) {
              // если это файл
```

```
                if($file !='.' && $file !='..'){
                        // то удаляем его
                        if(unlink($folderPath.'/'.$file)){
                                // вывод имени
                                // удаленного файла
                                echo "
                                   File: $file removed<br/>
                                ";
                                $count ++;
                        }
                }
        }
        // закрываем папку
        closedir($dir);
   }
}

// выводим количество удаленных файлов
echo "Count remove: $count";
```

Как правило, скрипты, подобные тому, что приведен в примере, устанавливаются на cron – планировщик задач. К примеру, скрипт автоматически вызывается раз в сутки и удаляет все временные файлы.

15. Проверка существования удаленных файлов

Не всегда приходится работать с файлами, которые находятся на сервере, с которого запускаются скрипты. Существует немало примеров, когда работать приходится с файлами, которые расположены на удаленных серверах.

Прежде чем получать данные с удаленных файлов, будет не лишним осуществлять проверку существования таких файлов. В связи с этим, следующий рецепт.

Пример:

```php
// пример использования
$result = isset_file("
      http://vk-book.ru/img/favicon.png
");
var_dump($result);


/**
* Функция определяет существование удаленного файла
*
* @param $url - string ссылка на файл
*
* @return bool
*/
function isset_file($url){
  $headers = @get_headers($url);
  // проверяем ответ сервера
  if(preg_match("|200|", $headers[0])){
        // если ответ с кодом 200
        return true;
  }else{
        return false;
  }
}
```

Приведенный пример, работает благодаря функции **get_headers**, которая получает заголовки отданные сервером, при обращении по url. В случае существования файла, сервер должен вернуть код ответа, равный 200. В противном случае код будет 404 или любой другой, в зависимости от настроек сервера.

16. Скачать и сохранить файл с сайта

После проверки существования удаленного файла, описанной в предыдущем примере, файл можно скачать и сохранить к себе на сервер – об этом и будет рецепт.

Для получения удаленного файла воспользуемся функцией **file_get_contents**. А для сохранения его на сервер - функцией **file_put_contents**.

Пример:

```
// два примера использования
getAndSaveFile(
   "http://vk-book.ru/img/favicon.png", './files/'
);

getAndSaveFile(
   "http://vk-book.ru/img/favicon.png",
   './files/', 'name_file.png'
);


/**
* Получить и сохранить файл
*
* @param string $url - ссылка на файл
* @param string $path - путь для сохранения файла
* @param string $fileName - не обязательный параметр,
* имя файла для сохранения
*
* @return bool - результат
*/
function getAndSaveFile($url, $path, $fileName=false)
{
   // открываем содержимое файла
   $file = @file_get_contents($url);
   // если не удалось получить файл, вернем false
   if(!$file) return false;
   // получаем имя файла, если не задано,
   // то берем из урла
   if(!$fileName) $fileName = basename($url);
   // сохраняем файл
   $resultSave = @file_put_contents(
         $path . $fileName, $file
   );
   // проверяем результат сохранения
   if ($resultSave || $resultSave > 0) return true;
```

```
    return false;
}
```

В примере, помимо функций для получения и сохранения файла, была использована функция – **basename**, с помощью которой можно получить имя и расширение файла из указанного пути.

17. Сохранение файла на компьютер пользователя

Еще одним немало важным моментом в работе с файлами является реализация предложения пользователю сохранить файл на компьютер. То есть при переходе по ссылке, для пользователя в браузере должно появиться окно с предложением сохранить файл. Этого не сложно добиться, задав определенные заголовки для браузера. Единственная сложность, возникающая в процессе реализации – это задать заголовки кроссбраузерно. Для всех адекватных браузеров заголовки задаются одни и те же, а вот для Internet Explorer уже совсем другие.

В рецепте приведен пример сохранения картинки в формате png, для файлов в другом формате следует заменить соответствующие заголовки.

Пример:

```
// имя файла, с которым он будет сохранен
$file_name = "pic.png";
// путь до файла
$file_path = "/images/pic.png";

// код 200, все хорошо
header("HTTP/1.1 200 OK");
header("Content-type: image/png"); // тип файла
// дата по Гринвичу
```

```
header('Expires: ' . gmdate('D, d M Y H:i:s') .
   ' GMT');
// определяем браузер
$ua = (isset($_SERVER['HTTP_USER_AGENT']))
         ? $_SERVER['HTTP_USER_AGENT'] : '';
$isMSIE = preg_match('@MSIE ([0-9].[0-9]{1,2})@'
   , $ua);
if ($isMSIE){
   // если это Internet Explorer
   // объясняем браузеру, что выводим файл
   header('
         Content-Disposition: attachment; filename="'
         . $file_name . '"'
   );
   header('
         Cache-Control: must-revalidate, post-check=0
         , pre-check=0'
   );
   header('Pragma: public');
}else{
   // если это НЕ Internet Explorer
   // объясняем браузеру, что выводим файл
   header('
         Content-Disposition: attachment;
         filename="' . $file_name . '"'
   );
   header('Pragma: no-cache');
}
// вывод файла в браузере
readfile($file_path);
```

18. Ini файлы. Что это такое и как их использовать?

Сначала пару слов о ini файлах. Что это такое? Это обыкновенный текстовый файл, который служит хранилищем для различных конфигураций. Такие файлы имеют определенную структуру, вот такого вида:

```
   ; файл config.ini
```

```
[db]
login = root
pass = 12345

[email]
admin = admin@test.com
support = support@test.com
```

db и email — это заголовки разделов. login, pass, admin и support — это параметры, с помощь знака равно им задается значение. В ini файле можно использовать комментарии, писать их нужно после знака точка с запятой .

Для чего нужны ini-файлы? В них можно хранить любую информацию, которую приходится часто менять или доступ, к которой должен быть простым. Например, это может быть настройки для подключения к базе данных.

В php можно быстро и просто получить значение параметра из ini-файла. Сделать это можно следующим образом:

```
// Получаем все параметры конфига вместе с разделами
$arrayAllConfig = parse_ini_file("config.ini", true);
// Выводим массив, в котором хранится login и pass
var_dump($arrayAllConfig[db]);
// Выводим login
var_dump($arrayAllConfig[db][login]);

// Получаем все параметры без разделов
$arrayAllConfig = parse_ini_file("config.ini");
// Выводим pass
var_dump($arrayAllConfig[pass]);
```

В приведенном выше примере используется функция **parse_ini_file**. Она позволяет считывать конфигурационный файл.

19. Выгрузка данных в Excel. Создание csv файлов

Зачастую в процессе разработке появляется необходимость выгрузки данных. Одним из самых удобных и читабельных способов является запись данных в таблицу excel. Excel файл может иметь несколько расширений – xls, xlsx, csv и тд. Самым удобным форматом с точки зрения генерации является csv, поскольку он устроен гораздо проще других. В таком файле столбцы таблицы разделяются каким-либо символом, который потом можно указать при открытии файла с помощью MS Office. По умолчанию символом-разделителем является точка с запятой, его и стоит использовать во избежание проблем при дальнейшем чтении файла. А для разделения строк в csv файлах используется перенос строки. Таким образом, получается, что для создания excel файла, достаточно создать текстовый файл, используя в нужных местах необходимые разделители столбцов и строк. Еще одним важным моментом, при генерации csv файлов, является кодировка файла. Для корректного отображения кириллицы следует использовать windows-1251.

Пример:

```
// массив имитирует данные,
// полученные, например, из базы данных
$data = array(
  array(
        'строка1 столбец1',
        'строка1 столбец2',
        'строка1 столбец3'
  ),
  array(
        'строка2 столбец1',
        'строка2 столбец2',
        'строка2 столбец3'
  ),
  array(
        'строка3 столбец1',
        'строка3 столбец2',
        'строка3 столбец3'
```

```
    )
);

// строка, которая будет записана в csv файл
$str = '';
// перебираем все данные
foreach($data as $value){
   $str .=
$value[0].';'.$value[1].';'.$value[2].";\r\n";
}
// задаем кодировку windows-1251 для строки
$str = iconv("UTF-8", "WINDOWS-1251",  $str);
// создаем csv файл и записываем в него строку
file_put_contents('test.csv', $str);
```

В приведенном примере, использовалась ранее не описанная функция - **iconv**. Она позволяет преобразовывать текст в нужную кодировку.

20. Разбор Excel таблицы. Получение данных из csv файла

В предыдущем рецепте, был приведен пример выгрузки данных в csv файл. Но бывает, что требуется реализовать обратную операцию – сделать разбор csv файла, считать хранящиеся в нем данные. Получение данных из csv файла можно реализовать несколькими разными способами: получить содержимое файла и разбить его с помощью функции **explode**. А можно использовать специальную php функцию – **fgetcsv**, которая читает строку и производит ее разбор. Второй вариант более красивый и корректный, поэтому приведенный ниже пример будет использовать именно этот способ.

Пример:

```
$row = 1;
// открываем файл
$file = fopen("test.csv", "r");
```

```
// разбираем файл построчно
while ((data = fgetcsv($file, 1000, ";")) !== false){
   // количество полей в строке
   $countPlace = count($data);
   echo "$countPlace полей в строке $row: <br/>";
   // выводим значения
   for ($c=0; $c < $countPlace; $c++) {
      echo $data[$c] . "<br/>";
   }
   $row++;
}
fclose($file);
```

21. Работа с zip архивами. Запаковка файлов

Иногда бывает необходимо программно запаковать файлы в zip архив. Для этих целей отлично подходит модуль - ZipArchive, который практически на всех серверах установлен вместе с самим php. Если же данный модуль не установлен, то следует это сделать. Для работы с модулем в php имеется класс – ZipArchive. Именно он и будет использован в рецепте, показывающем как можно запаковать все файлы и подпапки внутри выбранной папки.

Пример:

```
// путь к папке, файлы которой будем архивировать
$pathdir='test/';
//название архива
$nameArhive = 'test.zip';
// класс для работы с архивами
$zip = new ZipArchive;
// создаем архив, если все прошло удачно продолжаем
if (
   $zip->open($nameArhive, ZipArchive::CREATE) === true
){
   // открываем папку с файлами
   $dir = opendir($pathdir);
     // перебираем все файлы из нашей папки
    while($file = readdir($dir)){
```

```
            // проверяем файл ли мы взяли из папки
            if (is_file($pathdir.$file)){
                  // архивируем
                  $zip->addFile($pathdir.$file, $file);
                  // выводим название
                  // заархивированного файла
                  echo "Заархивирован: "
                  . $pathdir . $file . '<br/>';
            }
      }
      $zip->close(); // закрываем архив.
      echo 'Архив успешно создан';
}else{
      die ('Произошла ошибка при создании архива');
}
```

22. Работа с zip архивами. Распаковка файлов

В предыдущем рецепте, был приведён пример запаковки файлов в архив. Теперь будет приведен способ распаковки zip архивов. Как и при запаковке, в коде использован модуль и класс ZipArchive.

Пример:

```
// путь к папке, в которую будет распакован архив
$pathdir = 'test/';
//название архива
$nameArhive = 'test.zip';
// класс для работы с архивами
$zip = new ZipArchive;
// открываем архив
if ($zip->open($nameArhive) === true){
      // распаковываем архив
      $zip->extractTo($pathdir);
      // закрываем архив.
      $zip->close();
      echo 'Архив распакован в ' . $pathdir;
}else{
      die ('Произошла ошибка при распаковке архива');
}
```

23. Простое сжатие CSS файлов

Для больших проектов, которые имеют много css файлов большого размера, сжатие файлов стилей может немного ускорить загрузку страниц. Почти в каждом css файле можно найти куски закомментированного кода, повторы пробелов, табуляцию — все это можно удалить из файлов, а так же можно избавиться от переходов на новую строку. Помимо удаления лишних символов, будет полезно собрать все файлы в один файл, это тоже ускоряет загрузку. Все эти действия наглядно продемонстрирует ниже приведенный рецепт.

Пример:

```
// массив с путями до css файлов
$css_array = array(
    'css/style_1.css',
    'css/style_2.css'
);
// путь, куда будет сохранен сжатый файл
$new_file = "css/compression_file.css";
// вызываем функцию сжатия
$result = compression_files($css_array, $new_file);
var_dump($result); // вывод результата

/**
*   Функция для сжатия CSS файлов
*   Удаляет комментарии, табуляцию,
*   переходы на новую строку и повторяющиеся пробелы
*   А также собирает все файлы в один
*
*   @var $files_css array  - массив путей
*   до css файлов, которые необходимо сжать
*
*   @var $new_file  string - путь, куда будет
*   сохранен сжатый файл
*
```

```
*    @return bool - результат
*/
function compression_files($files_css, $new_file) {
    // получаем содержимое всех css файлов
    $content_css = "";
    foreach($files_css as $one_file){
        $content_css .= @file_get_contents($one_file);
        // если какой-то из файлов
        // не получилось прочитать
        if(!$content_css) return false;
    }
    // удаляем комментарии
    $content_css = preg_replace(
        '!/\*[^*]*\*+([^/][^*]*\*+)*/!', '',
        $content_css
    );
    // удаляем табуляции и переходы на новую строку
    $content_css = str_replace(
        array("\r\n", "\r", "\n", "\t"), ' ',
        $content_css
    );
    // удаляем повторяющиеся пробелы
    $content_css = preg_replace(
        '/ {2,}/', ' ', $content_css
    );
    // сохраняем результат в файл
    $css_file = fopen ($new_file, "w+");
    fwrite($css_file, $content_css);
    $result_save = fclose($css_file);
    // вернем результат сохранения
    return $result_save;
}
```

24. Создание папки

Для создания папки можно воспользоваться функцией **mkdir**. При создании, можно сразу задать права доступа на директорию. Только хочу обратить внимание, что заданные таким образом права будут работать, только в операционных системах *nix, а в windows такие права будут проигнорированы.

Пример:

```
// создать папку
// путь и имя новой папки
$folder = '/new_folder';
// создание папки, где 0700 - права
$result = mkdir ($folder, 0700);
if($result){
   echo "Папка создана";
}else{
   echo "Папка НЕ создана";
}
```

25. Удаления папки

Удаление папки осуществляется с помощью функции **rmdir**. Для успешного выполнения этой функции необходимо, чтобы папка была пуста. Таким образом, получается, что для удаления папки сначала необходимо ее очистить.

Пример удаления пустой папки, тут все просто:

```
// удалить папку
$folder = '/new_folder'; // путь до папки
rmdir($path . $folder);  // удаление папки
```

Для удаления папки, содержащей файлы, необходимо немного больше кода. Как уже выше говорилось, сначала необходимо удалить все файлы и все подпапки, которые содержаться внутри удаляемой папки. Для того, чтобы очистить папки можно написать небольшую рекурсивную функцию. Которая будет перебирать все вложенные файлы и удалять. Если же функция встретит не файл, а папку, то должна будет зайти вовнутрь и уже там удалить все файлы, после чего удалить эту подпапку.

Пример:

```
// удаление папки со всеми
// вложенными файлами и подпапками

$folder = '/new_folder'; // имя новой папки
remove_folder($folder); // удаление

// рекурсивная функция
function remove_folder($folder) {
  // получаем все файлы из папки
  if ($files = glob($folder . "/*")) {
        // удаляем по одному
        foreach($files as $file) {
              if(is_dir($file)){
                    // если попалась папка,
                    // то удаляем ее
                    remove_folder($file);
              }else{
                    // если попался файл
                    unlink($file);
              }
        }
  }
  // удаляем пустую папку
  rmdir($folder);
}
```

26. Установка прав на папку

Как уже раньше говорилось, установить права на папку можно, сразу при ее создании. Но не всегда работать приходится с вновь созданными папками, поэтому следующий рецепт покажет, как можно задать права на ранее созданную папку. Обратите внимание, что данный способ будет проигнорирован операционной системой windows. В примере будет использована функция **chmod**, она отвечает за установку прав.

Пример:

```
$folder = '/new_folder'; // папка
chmod($folder, 0777); // установка прав, 0777 – права
```

27. Получение размера папки

Еще одним не маловажным рецептом для работы с папками является получение размера папки, со всеми вложенными файлами и подпапками. Для реализации данного примера, хорошо подходит возможность создания рекурсивных функций. При вызове такой функции происходит проверка всех вложенных файлов – определяется их размер. В случае, когда попадается вложенная папка – подпапка, осуществляется рекурсивный вызов функции, которая перебирает файлы уже во вложенной подпапке.

Пример:

```
// указываем путь до папки или файла
$dirname = '/folder/';
// заносим в переменную размер папки или файла
$size = dir_size($dirname);
echo $formSize;

// функция для просмотра всех подпапок
// и всех вложенных файлов
function dir_size($dirname) {
  $totalsize = 0; // общий размер
  // открываем папку
  if ($dirstream = @opendir($dirname)) {
    // перебираем все что находится внутри папки
    while(($filename = readdir($dirstream))!==false){
      if ($filename != "." && $filename != ".."){
        // если попался файл
        if (is_file($dirname . "/" . $filename)){
          $totalsize += filesize(
              $dirname . "/" . $filename
          );
        }
```

```
            // если попалась папка
            if(is_dir($dirname . "/" . $filename)){
              $totalsize += dir_size(
                  $dirname . "/" . $filename
                );
            }
        }
      }
    }
    // закрываем папку
    closedir($dirstream);
    // возвращаем результат
    return $totalsize;
}
```

В результате выполнения приведенного выше скрипта, мы получим размер папки в байтах. Чтобы сделать красивый, форматированный вывод размера – перевести в Кб, Мб и тд, можно воспользоваться функцией, которая была описана в рецепте «Размер файла. Перевод байт в КБ, Мб и тд».

28. Массовая замена текста в файлах

Этот рецепт может быть полезен, при работе с большим количеством текстовых файлов. Пример наглядно покажет, как можно перебрать все файлы, находящиеся в заданной папке, а также во всех вложенных папках. Так же рецепт хорошо демонстрирует работу рекурсивных функций.

Пример:

```
// пример использования
$oldText = 'old text'; // что меняем
```

```php
$newText = 'new text'; // на что меняем
$folderName = "./files"; // в какой папке меняем
replace_txt($folderName, $oldText, $newText);


/**
* Функция замены текста во всех файлах папки
*
* @param string $folderName - пусть до папки
* @param string $oldText - искомый текст
* @param string $newText - на что меняем текст
*/
function replace_txt($folderName, $oldText, $newText){
  // открываем текущую папку
  $dir = opendir($folderName);
  // перебираем папку
  // перебираем пока есть файлы
  while (($file = readdir($dir)) !== false){
    // если это не папка
    if($file != "." && $file != ".."){
      // если файл
      if(is_file($folderName."/".$file)){
        // открываем файл
        $contentFile = file_get_contents(
              $folderName."/".$file
        );
        // для работы с файлами в
        // кодировке windows-1251
        //$contentFile = iconv(
        //    "windows-1251", "utf-8", $contentFile
        //);
        // делаем замену в тексте
        $contentFile = str_replace(
              $oldText, $newText, $contentFile
        );
        // сохраняем изменения
        file_put_contents(
              $folderName."/".$file,$contentFile
        );
      }
      // если папка, то рекурсивно
      // вызываем replace_txt
      if(is_dir($folderName."/".$file)){
```

```
            replace_txt(
              $folderName."/".$file, $oldText, $newText
            );
          }
        }
      }
      // закрываем папку
      closedir($dir);
    }
```

29. Поиск файла в папке

Еще одним хорошим примером использования рекурсии, может послужить поиск файла по папке и всем вложенным подпапкам. Для поиска файла, достаточно перебрать все файлы во всех папках и сравнить название файла с искомым. И в случае совпадения вернуть путь до найденного файла.

Пример:

```
// пример использования
$folderName = "./files"; // в какой папке ищем
$fileName = "test.txt"; // что ищем
$result = search_file($folderName, $fileName);
if($result){
    echo $result;
}else{
    echo "Нет такого файла";
}

/**
* Поиск файла по имени во всех папках и подпапках
*
* @param string $folderName - пусть до папки
* @param string $fileName - искомый файл
*/
function search_file($folderName, $fileName){
  // открываем текущую папку
  $dir = opendir($folderName);
  // перебираем папку
```

```
    // перебираем пока есть файлы
    while (($file = readdir($dir)) !== false){
      if($file != "." && $file != ".."){
        // если файл проверяем имя
        if(is_file($folderName . "/" . $file)){
          // если имя файла искомое,
          // то вернем путь до него
          if($file == $fileName)
            return $folderName."/".$file;
        }
        // если папка, то рекурсивно
        // вызываем search_file
        if(is_dir($folderName . "/" . $file)){
          return search_file($folderName . "/" . $file,
          $fileName);
        }
      }
    }
    // закрываем папку
    closedir($dir);
}
```

VII. Графика

Большинство приведенных в этом разделе примеров работают с библиотекой GD, которая практически всегда устанавливается в комплекте с самим php. Но если же этого не сделано, то прежде чем начать изучать рецепты данного раздела, следует позаботиться об установке этой библиотеки.

1. Проверка формата картинки

Прежде чем, начать работу с картинками, стоит определить ее формат. Убедиться в том, что данное изображение возможно

редактировать. Поэтому раздел по работе с графикой, стоит начать именно с этой темы.

Ниже приведена функция, которая определяет расширение файла и сравнивает с массивом форматов. Которые удовлетворяют условия, при которых код обработки картинок будет выполнен без ошибок.

Пример:

```
// файл, который будет проверен
$file = "./test.png";
// массив валидных форматов
$validFormat = array('jpg', 'jpeg', 'gif', 'png');
// проверка файла на корректный формат
$resultFormat = checkValidFormat($file, $validFormat);
if($resultFormat){
   echo "Корректный формат";
}else{
   echo "Не корректный формат";
}

/**
* Проверка корректности формата файла
*
* @param string $file - имя файла или путь до файла
* @param array $validFormat - массив с
* корректными форматами
*
* @return boolean - результат проверки
*/
function checkValidFormat($file, $validFormat){
     // определяем формат файла
     $format = end(explode(".", $file));
     if(in_array(strtolower($format), $validFormat)){
         return true;
     }
     return false;
}
```

В приведенном примере, использовались ранее не описанная функция - **strtolower**. Она осуществляет перевод текста в нижний

регистр. Это сделано с целью привести полученный формат файла к одному виду с теми, что заданы в массиве корректных форматов.

2. Проверка размера картинки

Еще одним немаловажным критерием при работе с картинками является их размер. В зависимости от большого или маленького размера, могут быть выполнены различные действия. Например, отказ от обработки изображения с выводом соответствующего сообщения или же будет выполнено уменьшение изображения до допустимых пределов.
В приведенном ниже рецепте, осуществляется получение размера картинки и сравнение их с максимально допустимыми.

Пример:

```
// файл, который будет проверен
$file = "./test.jpg";
// массив валидных размеров
$validSize = array(
    'width' => 100, // ширина в пикселях
    'height' => 100 // высота в пикселях
);
// проверка
$resultSize = checkValidSize($file, $validSize);
if($resultSize){
  echo "Корректный размер файла";
}else{
  echo "Не корректный размер файла";
}

/**
* Проверка корректности размера файла
*
* @param string $file - имя файла или путь до файла
* @param array $validSize - массив с
* корректными размерами.
```

```
* array(
*    // максимально допустимая ширина
*    'width' => $width,
*    // максимально допустимая высота
*    'heigth' => $heigth
* )
*
* @return boolean - результат проверки
*/
function checkValidSize($file, $validSize){
     // получаем массив размеров картинки
     $sizeImg = @getimagesize($file);
     // если не удалось обработать картинку
     if(!$sizeImg) return false;
     // сравнение размеров картинки с валидными
     if($validSize['width'] >= $sizeImg[0] &&
          $validSize['height'] >= $sizeImg[1]){
          return true;
     }
     return false;
}
```

В примере была использована не описанная ранее функция – **getimagesize**. Она позволяет получать размер картинки – ширину и высоту. Данные возвращает в виде массива.

3. Проверка наличия библиотеки GD

В двух предыдущих рецептах осуществлялась проверка корректности изображений, с которыми в последствии предстоит работать. Но прежде чем, начать их обрабатывать стоит проверить наличие самой библиотеки GD, с помощью которой и производятся все манипуляции с изображениями. Хочу обратить внимание, что это не единственная библиотека, для работы с изображениями, но в этом сборнике рецептов все примеры будут использовать только GD.

Пример:

```
if(isset_GD()){
   echo "Библиотека установлена!";
}else{
   echo "Библиотека НЕ установлена!";
}

function isset_GD(){
   if (!extension_loaded('gd')) {
     if (!dl('gd.so')) {
       return false;
     }
   }
   return true;
}
```

В приведенном примере использовались ранее не описанные функции – **extension_loaded** и **dl**. Первая функция определяет, загружено расширение или нет, вторая загружает расширение.

4. Изменение размера картинки

Теперь все приготовления окончены и можно приступать непосредственно к работе с картинками. Первый рецепт покажет, как можно изменять размер картинки. Пропорционально увеличивать или уменьшать. Если новый размер картинки будет задан не пропорционально, то холст создастся по новым размерам, а картинка изменится до максимально допустимого, без потери пропорций.

Пример:

```
// исходное изображение
$source = "test.jpg";
// путь для сохранения новой картинки
$new_file = "test_new.jpg";
```

```
$width = 200; // новая ширина
$height = 200; // новая высота

// цвет заливки фона
$rgb = 0xffffff;
//узнаем размеры исходной картинки
$size = getimagesize($source);
//пропорция ширины
$x_ratio = $width / $size[0];
//пропорция высоты
$y_ratio = $height / $size[1];
// определяем соотношения ширины к высоте
$ratio = min($x_ratio, $y_ratio);
$use_x_ratio = ($x_ratio == $ratio);
// высчитываем новую ширину картинки
$new_width   = $use_x_ratio  ? $width  :
  floor($size[0] * $ratio);
// высчитываем новую высоту картинки
$new_height  = !$use_x_ratio ? $height :
  floor($size[1] * $ratio);
// расхождение с заданными параметрами по ширине
$new_left    = $use_x_ratio  ? 0 :
  floor(($width - $new_width) / 2);
// расхождение с заданными параметрами по высоте
$new_top     = !$use_x_ratio ? 0 :
  floor(($height - $new_height) / 2);
// создаем холст пропорциональное сжатой картинке
$img = imagecreatetruecolor($width,$height);
// заливаем холст цветом $rgb
imagefill($img, 0, 0, $rgb);
// загружаем исходную картинку
$photo = imagecreatefromjpeg($source);
// копируем на холст сжатую картинку с учетом
расхождений
imagecopyresampled($img, $photo, $new_left, $new_top,
  0, 0, $new_width, $new_height, $size[0], $size[1]);
// сохраняем результат
imagejpeg($img, $new_file);
// очищаем память после выполнения скрипта
imagedestroy($img);
imagedestroy($photo);
```

В этом рецепте появилось много, ранее не используемых функций. Обо всех по порядку:

Min – находит минимальное значение, из сравниваемых.

Imagecreatetruecolor – создает новое, пустое изображение заданного размера.

Imagefill – осуществляет заливку заданным цветом. В данном примере заливается все изображение.

Imagecreatefromjpeg – создает изображение из файла, работает только с форматом jpeg.

Imagecopyresampled – копирует одно изображение на другое, с возможностью смещения и изменения размера копируемого. В данном случае, копируемое изображение меняет размер и центруется, относительно пустого изображения.

Imagejpeg – Сохраняет или выводит в браузер изображение в формате jpeg.

Imagedestroy – очищает память занятую изображением.

5. Изменение размера PNG картинки

В этом рецепте будет приведен, немного модифицированный, предыдущий пример, который сможет изменять размер картинок в формате png без потери прозрачности.

Отличие этого рецепта от предыдущего заключается в двух функциях – загрузка изображения и сохранения. Функции использованы аналогичные, но поддерживающие работу с png -

imagecreatefrompng и **imagepng**. А также после создания пустого изображение, оно не заливается ни каким цветом, а наоборот делается прозрачным с помощью функций – **imagealphablending** и **imagesavealpha**.

Пример:

```
// исходное изображение
$source = "test_png.png";
// путь для сохранения новой картинки
$new_file = "test_png_new.png";
$width = 200; // новая ширина
$height = 200; // новая высота

//узнаем размеры исходной картинки
$size = getimagesize($source);
//пропорция ширины
$x_ratio = $width / $size[0];
//пропорция высоты
$y_ratio = $height / $size[1];
// определяем соотношения ширины к высоте
$ratio = min($x_ratio, $y_ratio);
$use_x_ratio = ($x_ratio == $ratio);
// высчитываем новую ширину картинки
$new_width   = $use_x_ratio  ? $width  :
  floor($size[0] * $ratio);
// высчитываем новую высоту картинки
$new_height  = !$use_x_ratio ? $height :
  floor($size[1] * $ratio);
// расхождение с заданными параметрами по ширине
$new_left    = $use_x_ratio  ? 0 :
  floor(($width - $new_width) / 2);
// расхождение с заданными параметрами по высоте
$new_top     = !$use_x_ratio ? 0 :
  floor(($height - $new_height) / 2);
// создаем холст пропорциональное сжатой картинке
$img = imagecreatetruecolor($width,$height);
// делаем его прозрачным
imagealphablending($img, false);
imagesavealpha($img, true);
// загружаем исходную картинку
$photo = imagecreatefrompng($source);
```

```
// копируем на холст сжатую картинку
// с учетом расхождений
imagecopyresampled($img, $photo, $new_left, $new_top,
 0, 0, $new_width, $new_height, $size[0], $size[1]);
// сохраняем результат
imagepng($img, $new_file);
// очищаем память после выполнения скрипта
imagedestroy($img);
imagedestroy($photo);
```

6. Получение фрагмента картинки

Рецепт наглядно показывает, как можно вырезать фрагмент изображения. Для примера используется картинка в формате jpeg, но при необходимости тоже самое можно проделать с любым другим форматом. Для получения части картинки необходимо знать координаты точки по оси X и Y, это будет верхний левый угол квадрата, который будет вырезан. Также потребуется размер вырезаемого фрагмента – ширина и высота.

Пример:

```
// исходное изображение
$source = "test.jpg";
// путь для сохранения новой картинки
$new_file = "test_new.jpg";
// координата по оси X, начало вырезаемого фрагмента
$x_pic = 100;
// координата по оси Y, начало вырезаемого фрагмента
$y_pic = 50;
$width = 200; // ширина фрагмента
$height = 200; // высота фрагмента

// создаем холст с размером равным,
// вырезаемому фрагменту
$img = imagecreatetruecolor($width, $height);
// загружаем исходную картику
$pic = imagecreatefromjpeg($source);
// копируем на холст нужную часть картинки
```

```
imagecopy(
   $img, $pic, 0, 0, $x_pic, $y_pic, $width, $height
);
// сохраняем результат
imagejpeg($img, $new_file);
// очищаем память после выполнения скрипта
imagedestroy($img);
imagedestroy($pic);
```

В примере была использована функция **imagecopy**, которая позволяет копировать одно изображение на другое, с возможностью смещения по осям X и Y.

7. Вывод изображения в браузере

В рассмотренных ранее рецептах, созданные изображения сохранялись в файл. Этот способ вывода используется часто, но иногда приходится отображать результат сразу в браузер, без предварительного сохранения.

Для ввода изображения в браузере достаточно функции **imagejpeg**(при работе с картинками в формате jpeg), предварительно передав браузеру заголовок, который задает тип контента: "Content-Type: image/jpeg".

Пример:

```
$width = 200; // ширина изображения
$height = 200; // высота изображения
// создаем изображение
$img = imagecreatetruecolor($width, $height);
// цвет заливки фона. зеленый
$rgb = 0x00ff00;
// заливаем холст цветом $rgb
imagefill($img, 0, 0, $rgb);
// заголовок для браузера
```

```
header("Content-Type: image/jpeg");
// выводим результат в браузер
imagejpeg($img);
// очищаем память после выполнения скрипта
imagedestroy($img);
imagedestroy($pic);
```

8. Универсальные функции создания и сохранения картинок

Как, наверняка, уже стало ясно из предыдущих рецептов, для работы с разными форматами изображений, приходится использовать разные php функции при создании и сохранении/выводе в браузер изображения. Например, при сохранении картинки в формате jpg используется функция **imagejpeg**, для png - **imagepng** , для gif - **imagegif**. Так же обстоят дела и с функциями создания изображения разных форматов. Это создает некоторые проблемы в случаях, когда на обработку может попасть картинка любого формата. Ниже приведенный рецепт поможет решить сложившуюся проблему.

Пример:

```
$source = "test.jpg";
// $source = "test.jpeg";
// $source = "test.png";
// $source = "test.gif";
// получаем функцию, для создания изображения
$imageCreateFunc = get_image_create_func($source);
// получаем функцию, для сохранения/вывода изображения
$imageSaveFunc = get_image_save_func($source);

// загружаем исходную картинку
$pic = $imageCreateFunc($source);
// заголовок для браузера
header("Content-Type: image/jpeg");
```

```php
// выводим результат в браузер
$imageSaveFunc($pic);
// очищаем пямять
imagedestroy($pic);

// получаем функцию, для создания изображения
function get_image_create_func($source){
  // получаем данные о файле
  $size = @getimagesize($source);
  // если ни чего не получили
  if(!$size) return false;
  // определяем формат картинки
  $format = strtolower(substr($size['mime'],
        strpos($size['mime'], '/') + 1)
  );
  // при необходимости, меняем jpg на jpeg
  if($format == "jpg") $format = "jpeg";
  // создаем функцию, для полученного
  // формата изображения
  $func = "imagecreatefrom" . $format;
  // проверяем существование полученной функции
  if(!function_exists($func)) return false;
  // возвращаем результат
  return $func;
}

// получаем функцию, для сохранения/вывода изображения
function get_image_save_func($source){
  $size = @getimagesize($source);
  if(!$size) return false;
  $format = strtolower(
    substr($size['mime'], strpos($size['mime'],
    '/')+1)
  );
  if($format == "jpg") $format = "jpeg";
  $func = "image" . $format;
  if(!function_exists($func)) return false;
  return $func;
}
```

В приведенном примере использовалась ранее не описанная функция – **function_exists**. Она проверяет, определена функция или нет.

9. Поворот изображения

Поворот изображения реализуется с помощью функции **imagerotate**. Угол поворота задается в градусах(0-360), его можно задавать как положительным, так и отрицательным значением. От знака градуса зависит, в какую сторону будет сделан поворот, по часовой стрелке или против нее.

Пример:

```
// картинка
$source = "test.jpg";
// угол, на который будет осуществлен поворот
$degree = 45;

// Загрузка изображения
$img = imagecreatefromjpeg($source);
// Поворот. Пустые углы заливаем цветом 0xffffff
$rotate_img = imagerotate($img, $degree, '0xffffff');
// заголовок для браузера
header("Content-Type: image/jpeg");
// выводим результат в браузер
imagejpeg($rotate_img);
// очищаем пямять
imagedestroy($rotate_img);
```

10. Поворот PNG изображения с сохранением прозрачности

Поворот png изображения, которое имеет прозрачность, немного сложней поворота обычной картинки. Необходимо при повороте, задать прозрачный фон, который создается с помощью функции **imagecolorallocatealpha**. А также после самого поворота нужно задать прозрачность для повернутого изображения.

Пример:

```
// картинка
$source = "test.png";
// угол, на который будет осуществлен поворот
$degree = 45;

// Загрузка изображения
$img = imagecreatefrompng($source);
// создаем прозрачный фон
$bg = imagecolorallocatealpha($img, 0, 0, 0, 127);
// поворот на нужный угол
$rotate_img = imagerotate($img, $degree, $bg);
// задаем прозрачность для повернутой картинки
imagesavealpha($rotate_img, true);

// заголовок для браузера
header("Content-Type: image/png");
// выводим результат в браузер
imagepng($rotate_img);
// очищаем пямять
imagedestroy($img);
imagedestroy($rotate_img);
```

11. Рисование линии. Стиль, цвет, толщина

Помимо обработки картинок, с помощью GD можно рисовать простые элементы. В этом рецепте приведен пример нанесения линий. Рисовать можно как на пустых холстах, так и на загруженных изображениях. Линии могут иметь различный цвет, толщину и стиль отображения – не прерывная, пунктир, точки, точка и тире и так

далее. Для установки стиля, используется функция **imagesetstyle**, которая принимает массив с порядком нанесения и цветом пикселей. Чтобы определить толщину линии применяется функция – **imagesetthicknes**. А для рисования самой линии используется функция **imageline**,которая принимает значение начальных и конечных координат по осям X и Y.

Пример:

```
$width = 200; // ширина изображения
$height = 200; // высота изображения
// создаем изображение, на котором будем рисовать
$img = imagecreatetruecolor($width, $height);
// цвет заливки фона
$rgb = 0xFFFFFF;
// заливаем холст цветом $rgb
imagefill($img, 0, 0, $rgb);

// создаем цвета
$red   = imagecolorallocate($img, 255, 0, 0);
$white = imagecolorallocate($img, 255, 255, 255);
$black = imagecolorallocate($img, 0, 0, 0);

// координаты линии
$x1 = 20; $y1 = 50; $x2 = 180; $y2 = 50;
// рисуем обычную линию
imageline ($img, $x1, $y1, $x2, $y2, $black);

// установка стиля линии (пунктир)
// 4 белых пикселя, 4 черных
$style = array(
   $white, $white, $white, $white,
   $black, $black, $black, $black
);
imagesetstyle($img, $style);
// координаты линии
$x1 = 20; $y1 = 150; $x2 = 180; $y2 = 50;
// рисуем пунктирную линию
imageline (
   $img, $x1, $y1, $x2, $y2, IMG_COLOR_STYLED
);
```

```
// толщина линии
imagesetthickness($img, 5);
// координаты линии
$x1 = 20; $y1 = 150; $x2 = 180; $y2 = 150;
// рисуем толстую линию
imageline ($img, $x1, $y1, $x2, $y2, $red);

// заголовок для браузера
header("Content-Type: image/png");
// выводим результат в браузер
imagepng($img);
// очищаем память после выполнения скрипта
imagedestroy($img);
```

Помимо уже описанных выше функций, в примере была использована еще одна - **imagecolorallocate**. С ее помощью можно создавать цвета, для дальнейшего использования при работе с изображением.

12. Рисование прямоугольников и квадратов

Нарисовать прямоугольник можно несколькими способами. Первый – воспользовавшись предыдущим рецептом, нарисовать четыре линии. Второй – использовать специальные функции: **imagerectangle**, которая рисует квадрат по координатам двух, расположенных по диагонали улов. При рисовании прямоугольника используются координаты верхнего левого и нижнего правого угла. Также есть еще одна функция - **imagefilledrectangle**, она схожа с первой, только рисует закрашенный прямоугольник. Используя две эти функции, можно так же легко нарисовать и квадраты, главное, чтобы стороны были равны.

Пример:

```php
$width = 200; // ширина изображения
$height = 300; // высота изображения
// создаем изображение, на котором будем рисовать
$img = imagecreatetruecolor($width, $height);
// цвет заливки фона
$rgb = 0xFFFFFF;
// заливаем холст цветом $rgb
imagefill($img, 0, 0, $rgb);

// определяем цвета
$red = imagecolorallocate($img, 255, 0, 0);
$green = imagecolorallocate($img, 0, 255, 0);
$blue = imagecolorallocate($img, 0, 0, 255);

// толщина линий
imagesetthickness($img, 3);

// рисуем прямоугольник красными линиями
$x1 = 20; $y1 = 40; $x2 = 150; $y2 = 90;
imagerectangle($img, $x1, $y1, $x2, $y2, $red);

// рисуем закрашенный зеленый прямоугольник
$x1 = 20; $y1 = 110; $x2 = 150; $y2 = 160;
imagefilledrectangle($img, $x1, $y1, $x2, $y2,
$green);

// рисуем синий квадрат
$x1 = 60; $y1 = 180; $x2 = 110; $y2 = 230;
imagefilledrectangle($img, $x1, $y1, $x2, $y2, $blue);

// заголовок для браузера
header("Content-Type: image/png");
// выводим результат в браузер
imagepng($img);
// очищаем память после выполнения скрипта
imagedestroy($img);
```

13. Рисование окружностей, эллипсов и дуг

Для рисования эллипсов существует функция **imageellipse**, которая принимает несколько параметров: центр эллипса по осям X

и Y, ширина и высота, цвет. Поскольку растянутость и сжатие можно регулировать параметрами ширины и высоты, рисование эллипсов происходит очень просто. Эту же функцию можно использовать при нанесении на изображение окружности, главное, чтобы ширина и высота были одинаковыми.

С помощью аналогичной функции **imagefilledellipse**, можно рисовать эллипсы и окружности, залитые выбранным цветом.

Для рисования дуги, описанные выше функции, не подходят, в этом случае следует использовать **imagearc**. Эта функция принимает также несколько параметров: центр эллипса по осям X и Y, ширина и высота, угол начала и угол окончания дуги, цвет.

Пример:

```
$width = 200; // ширина изображения
$height = 250; // высота изображения
// создаем изображение, на котором будем рисовать
$img = imagecreatetruecolor($width, $height);
// цвет заливки фона
$rgb = 0xFFFFFF;
// заливаем холст цветом $rgb
imagefill($img, 0, 0, $rgb);

// определяем цвета
$red = imagecolorallocate($img, 255, 0, 0);
$green = imagecolorallocate($img, 0, 255, 0);
$blue = imagecolorallocate($img, 0, 0, 255);
$black = imagecolorallocate($img, 0, 0, 0);

// координаты центра
$x = 100;
$y = 50;
// радиус
$radius = 30;
// Рисуем красный круг
imageellipse(
   $img, $x, $y, $radius * 2, $radius * 2, $red
```

```
);

// координаты центра
$x = 100;
$y = 120;
// радиус
$radius = 30;
// Рисуем закрашенный зеленый круг
imagefilledellipse(
  $img, $x, $y, $radius * 2, $radius * 2, $green
);

// координаты центра
$x = 100;
$y = 175;
// ширина
$width = 60;
// высота
$height = 30;
// Рисуем закрашенный синий эллипс
imagefilledellipse(
  $img, $x, $y, $width, $height, $blue
);

// координаты центра
$x = 100;
$y = 200;
// ширина
$width = 100;
// высота
$height = 60;
// угол начала дуги
$degree_start = 0;
// угол окончания дуги
$degree_end = 180;
// рисуем черную дугу
imagearc(
  $img, $x, $y, $width, $height,
  $degree_start, $degree_end, $black
);

// заголовок для браузера
header("Content-Type: image/png");
```

```
// выводим результат в браузер
imagepng($img);
// очищаем память после выполнения скрипта
imagedestroy($img);
```

14. Нанесение текста на изображение. 1 способ

Помимо рисования графических элементов, в GD есть возможность нанесения текста на изображение. Причем несколькими способами. Первый, простой, описан в этом рецепте, написание текста реализуется с помощью функции **imagestring**. Она может наносить текст, используя стандартные шрифты. Стандартных шрифтов всего пять, они задаются числом от одного до пяти. Чем больше значение, тем крупнее шрифт. Также функция может принимать цвет текста и координаты начала строки по осям X и Y.

Хочу обратить внимание, что у этого способа нанесения текста имеется один большой недостаток, функция **imagestring** использует стандартные шрифты в кодировке latin2. В связи с этим, вывод кириллицы не возможен.

Пример:

```
$width = 200; // ширина изображения
$height = 200; // высота изображения
// создаем изображение, на котором будем рисовать
$img = imagecreatetruecolor($width, $height);
// цвет заливки фона
$rgb = 0xFFFFFF;
// заливаем холст цветом $rgb
imagefill($img, 0, 0, $rgb);

// определяем цвет
$red = imagecolorallocate($img, 255, 0, 0);

// координаты начала текста
```

```
$x = 30; $y = 30;
// выбираем шрифт, число от 1 до 5
$font = 3;
imagestring(
   $img, $font, $x, $y, 'My first text', $red
);

// заголовок для браузера
header("Content-Type: image/png");
// выводим результат в браузер
imagepng($img);
// очищаем память после выполнения скрипта
imagedestroy($img);
```

15. Нанесение текста на изображение. 2 способ

В предыдущем рецепте, уже затрагивалась тема нанесения текста, только в примере присутствовал один серьезный недостаток – отсутствовала возможность нанесения на изображение кириллического текста. Этот рецепт исправит недостаток предыдущего. Для нанесения текста в любой кодировке и в любом языке возможно при использовании сторонних шрифтов. В реализации примера используется функция - **imagettftext**. Она может принимать несколько значений: размер текста в пикселях, цвет, начало текста относительно координат по осям X и Y, сам наносимый текст и самое главное, файл шрифта, который позволит наносить текст в нужной кодировке.

Пример:

```
$width = 200; // ширина изображения
$height = 100; // высота изображения
// создаем изображение, на котором будем рисовать
$img = imagecreatetruecolor($width, $height);
// цвет заливки фона
$rgb = 0xFFFFFF;
```

```php
// заливаем холст цветом $rgb
imagefill($img, 0, 0, $rgb);

// путь к шрифту
$fontName = "impact.ttf";
// размер шрифта
$fontSise = 18;
$x = 50; // отступ слева
$y = 50; // отступ справа
// текст, который будем наносить на картинку
// \n обозначает переход на новую строку
$text = "Текст кир\nиллица";
$textColor = 0x000000; // цвет шрифта
// нанесение текста
imagettftext(
  $img, $fontSise, 0, $x, $y,
  $textColor, $fontName, $text
);

// заголовок для браузера
header("Content-Type: image/png");
// выводим результат в браузер
imagepng($img);
// очищаем память после выполнения скрипта
imagedestroy($img);
```

16. Нанесение на изображение текста с обводкой

Для нанесения на изображение текста с обводкой специальных функций нет. Но из этого положения можно выйти, используя функцию - **imagettftext**. Все что нужно сделать, это вывести текст не один раз, как в стандартном случае, а девять. Да, девять раз, из них восемь раз вывод текста нужно делать цветом обводки, постоянно смещая текст 1-2 пикселя влево, вправо, вниз, вверх, влево и вверх, вправо и вверх, вправо и вниз, влево и вниз. И девятый раз необходимо выводить текст уже тем цветом, которого он должен быть.

Пример:

```
$width = 200; // ширина изображения
$height = 100; // высота изображения
// создаем изображение, на котором будем рисовать
$img = imagecreatetruecolor($width, $height);
// цвет заливки фона
$rgb = 0xFFFFFF;
// заливаем холст цветом $rgb
imagefill($img, 0, 0, $rgb);

// путь к шрифту
$fontName = "impact.ttf";
// размер шрифта
$fontSise = 18;
$x = 50; // отступ слева
$y = 50; // отступ справа
// текст, который будем наносить на картинку
// \n обозначает переход на новую строку
$text = "Текст кир\nиллица";

$textColor = 0xFFFFFF; // цвет шрифта
$aroundColor = 0x000000; // цвет обводки
// обводка текста
// смещение вправо
imagettftext(
   $img, $fontSise, 0, $x+2, $y,
   $aroundColor, $fontName, $text
);
// смещение влево
imagettftext(
   $img, $fontSise, 0, $x-2, $y,
   $aroundColor, $fontName, $text
);
// смещение вниз
imagettftext(
   $img, $fontSise, 0, $x, $y+2,
   $aroundColor, $fontName, $text
);
// смещение вверх
imagettftext(
   $img, $fontSise, 0, $x, $y-2,
   $aroundColor, $fontName, $text
```

```
);
// смещение вправо и вниз
imagettftext(
   $img, $fontSise, 0, $x+1, $y+1,
   $aroundColor, $fontName, $text
);
// смещение вправо и вверх
imagettftext(
   $img, $fontSise, 0, $x+1, $y-1,
   $aroundColor, $fontName, $text
);
// смещение влево и вверх
imagettftext(
   $img, $fontSise, 0, $x-1, $y-1,
   $aroundColor, $fontName, $text
);
// смещение влево и вниз
imagettftext(
   $img, $fontSise, 0, $x-1, $y+1,
   $aroundColor, $fontName, $text
);
// вывод самого текста
imagettftext(
   $img, $fontSise, 0, $x, $y,
   $textColor, $fontName, $text
);

// заголовок для браузера
header("Content-Type: image/png");
// выводим результат в браузер
imagepng($img);
// очищаем память после выполнения скрипта
imagedestroy($img);
```

17. Нанесение на изображение текста с подчеркиванием

При нанесении текста на изображение не редко случается необходимость, вывода подчеркнутого текста. К сожалению, для

этих целей стандартных функций нет. Но, реализовать подчеркивание можно. Для этого необходимо вывести текст и нарисовать под ним линию. Загвоздка заключается в том, что необходимо знать координаты нанесения линии, а также ее длину. Для определения начала линии можно воспользоваться координатами самого текста, только прибавить к значению по оси Y размер самого текста. А для получения длины, можно воспользоваться функцией **imagettfbbox**, которая определяет положение текста. Функция возвращает массив с восемью элементами, координаты углов, обрамляющей текст рамки. Используя полученные, координаты, можно легко высчитать длину текста.

Пример:

```
$width = 200; // ширина изображения
$height = 100; // высота изображения
// создаем изображение, на котором будем рисовать
$img = imagecreatetruecolor($width, $height);
// цвет заливки фона
$rgb = 0xFFFFFF;
// заливаем холст цветом $rgb
imagefill($img, 0, 0, $rgb);

// путь к шрифту
$fontName = "impact.ttf";
// размер шрифта
$fontSise = 18;
$x = 50; // отступ слева
$y = 50; // отступ справа
// текст, который будем наносить на картинку
$text = "Любой текст";
// цвет шрифта
$textColor = 0x000000;
// отступ линии от начала текста
$paddingLine = 3;

// выводим текст
imagettftext(
```

```
    $img, $fontSise, 0, $x, $y,
    $textColor, $fontName, $text
);
// определяем размер текста
$textbox = imagettfbbox(
    $fontSise, 0, $fontName, $text
);
// вычисляем ширину текста
$text_width = $textbox[2] - $textbox[0];
// толщина линии
imagesetthickness($img, 3);
// рисуем линию
imageline (
    $img, $x, $y + $paddingLine, $x + $text_width,
    $y + $paddingLine, $textColor
);

// заголовок для браузера
header("Content-Type: image/png");
// выводим результат в браузер
imagepng($img);
// очищаем память после выполнения скрипта
imagedestroy($img);
```

18. Создание картинки из текста. Защите e-mail от спама

Ни для кого не секрет, что сайты, особенно крупные, постоянно сканируются роботами, и не только поисковыми. Зачастую спамеры собирают свои базы e-mail адресов, сканируя сайты. Чтобы это предотвратить, можно выводить e-mail пользователей не в текстовом виде, а картинкой. Для этого необходимо предварительно перевести текст в картинку, единственная проблема, возникающая при этом, это динамический размер изображения, поскольку длина e-mail всегда разная. В решении поможет, описанная в предыдущем рецепте, функция - **imagettfbbox**.

Пример:

```
// шрифт
$font = 'impact.ttf';
// размер текста
$fontSize = 12;
// текст. e-mail
$imgText = "admin@vk-book.ru";
// определяем размер картинки в зависимости
// от длины и размера текста
$boxText = ImageTTFBBox($fontSize, 0, $font,
$imgText);
// размер будущей картинки
$widthImg = $boxText[2] - $boxText[0];
$heightImg = $boxText[1] - $boxText[7];
// создаем полотно
$img = imagecreatetruecolor($widthImg, $heightImg);
// задаем цвет фона
$fonColor = imagecolorallocate($img, 255, 255, 255);
// заливаем фон
imagefill($img, 0, 0, $fonColor);
// цвет текста
$textColor = imagecolorallocate ($img, 0, 0, 0);
// наносим текст
imagettftext (
   $img, $fontSize, 0, 0, $fontSize,
   $textColor, $font, $imgText
);
// выводим картинку в браузере
header("Content-type: image/png");
imagePng($img);
// очищаем память
imageDestroy($img);
```

19. Наложения водяного знака на картинку

Водяным знаком может служить текст или картинка. Как накладывать текст, можно почитать в предыдущих рецептах, а как наложить на одно изображение другое, описано ниже.

При наложении водяных знаков, накладываемой картинке задается прозрачность 40-60%.

В примере, приведенном ниже, водяному знаку будет задана прозрачность 60% и размещен он будет в нижний правый угол.

Пример:

```
// исходная картинка
$img = "test.jpg";
// получаем размер картинки
$size = getimagesize($img);
$height = $size[1]; // высота
$width = $size[0]; // ширина

// картинка, которая будет использована
// в качестве водяного знака
$watermark_src = 'watermark.png';
// получаем размер водяного знака
$sizeWM = getimagesize($watermark_src);
$heightWM = $sizeWM[1]; // высота водяного знака
$widthWM = $sizeWM[0]; // ширина водяного знака
// задаем прозрачность водяного знака
$opacity = 60;

//Загружаем изображения
$image = imagecreatefromjpeg($img);
$watermark = imagecreatefrompng($watermark_src);

// высчитываем координаты, для водяного знака.
// Внизу справа
$x = $width - $widthWM;
$y = $height - $heightWM;

//Копируем водяной знак на изображение
imagecopymerge(
   $image, $watermark, $x, $y, 0, 0,
   $widthWM, $heightWM, $opacity
);

// задаем заголовок, чтоб вывести результат в браузере
header('Content-Type: image/jpeg');
```

```
// выводим картинку
imagejpeg($image);
// очищаем память
imagedestroy($image);
imagedestroy($watermark);
```

В примере, была использована, ранее не описанная функция - **imagecopymerge**. Она позволяет копировать часть изображения и накладывать его на другое изображение. А также при наложении есть возможность задавать прозрачность, для копируемого изображения.

20. Наложение PNG картинок с прозрачностью

В прошлом примере, наложение происходило без учета прозрачности, накладываемой картинки. Даже если бы прозрачность и существовала, то вместо нее бы появился белый цвет. В этом рецепте мы исправим этот недочет. Поможет в этом функция - **imagecopy**, которая копирует одно изображение на другое.

Пример:

```
// исходная картинка
$img = "test.jpg";

// накладываемая картинка
$watermark_src = 'watermark.png';
// получаем ее размер
$sizeWM = getimagesize($watermark_src);
$heightWM = $sizeWM[1]; // высота
$widthWM = $sizeWM[0]; // ширина

// Загружаем изображения
$image = imagecreatefromjpeg($img);
$watermark = imagecreatefrompng($watermark_src);
// задаем прозрачность
imagesavealpha($watermark, true);
```

```
// координаты верхнего левого угла накладываемой
картинки
$x = 50;
$y = 50;

// Копируем
imagecopy(
   $image, $watermark, $x, $y, 0, 0,
   $widthWM, $heightWM
);

// задаем заголовок, чтоб вывести результат в браузере
header('Content-Type: image/jpeg');
// выводим картинку
imagejpeg($image);
// очищаем память
imagedestroy($image);
imagedestroy($watermark);
```

21. Наложение маски на изображение

При наложении маски исходное изображение будет обрезано по форме маски, в роли которой может любая картинка, допустимого формата – jpeg, jpg, png, gif. Картинка-маска должна иметь однотонный цвет в области, которая будет сохранена, например черный. А все остальное, что будет удалено, должно иметь любой другой цвет, главное не черный.

Таким образом, имея две картинки – исходную и маску, мы можем сделать наложение. Для наложения маски нет специальных функций, но можно попиксельно перенести исходную картинку на чистый холст, предварительно проверяя цвет пикселя у картинки-маски по тем же координатам.

Пример:

```php
// исходная картинка
$image = "test.jpg";
// картинка маска
$mask = "mask.png";

// загружаем исходную картинку
$image = imagecreatefromjpeg($image);
// загружаем маску
$mask = imagecreatefrompng($mask);
// определяем ширину картинки
$width =  imagesx($image);
// определяем высоту картинки
$height = imagesy($image);
// определяем ширину маски
$m_width =  imagesx($mask);
// определяем высоту маски
$m_height = imagesy($mask);

// создаем холст для будущей картинки
$img = imagecreatetruecolor($width, $height);
// определяем прозрачный цвет для картинки. Черный
$transColor = imagecolorallocate($img, 0, 0, 0);
// задаем прозрачность для картинки
imagecolortransparent($img, $transColor);

// перебираем исходную картинку по пикселю
for($posX = 0; $posX < $width; $posX++){
  for($posY = 0; $posY < $height; $posY++){
    // получаем индекс цвета пикселя
    // в координате $posX, $posY для картинки
    $colorIndex = imagecolorat($image, $posX, $posY);
    // получаем цвет по его индексу в формате RGB
    $colorImage = imagecolorsforindex(
        $image, $colorIndex
    );
    // получаем индекс цвета пикселя
    // в координате $posX, $posY для маски
    $colorIndex = imagecolorat($mask, $posX, $posY);
    // получаем цвет по его индексу в формате RGB
    $maskColor = imagecolorsforindex(
        $mask, $colorIndex
    );
    // если в точке $posX, $posY цвет маски черный,
```

```
        // то наносим на холст пиксель с нужным цветом
        if (
            // проверка пикселя на черный цвет
            $maskColor['red'] == 0 AND
            $maskColor['green'] == 0 AND
            $maskColor['blue'] == 0 AND
            // если размер маски меньше исходной картинки,
            // то за ее пределами тоже ни чего не рисуем
            $m_width > $posX AND
            $m_height > $posY
         ){
            // получаем цвет для пикселя
            $colorIndex = imagecolorallocate(
                  $img,
                  $colorImage['red'],
                  $colorImage['green'],
                  $colorImage['blue']
            );
            // рисуем пиксель
            imagesetpixel(
                  $img, $posX, $posY, $colorIndex
            );
        }
    }
}

// заголовок для браузера
header('Content-type: image/png');
// выводим картинку в браузере
imagepng($img);
// чистим память
imagedestroy($img);
```

В рецепте использовалось несколько ранее не описанных функций, о каждой по порядку.

Imagesx и **imagesy** – получают ширину и высоту изображения.

imagecolortransparent – определяет выбранный цвет как прозрачный.

imagecolorat – получает индекс цвет а пикселя по заданным координатам.

imagecolorsforindex – получает цвета в формате RGBA, соответствующие индексу.

imagesetpixel – рисует пиксель по заданным координатам.

22. Закругление углов картинки

Стандартных функций, которые выполняют закругление углов, не существует. Поэтому необходимо самим разрабатывать алгоритм для таких случаев. Один из способов реализации закругления углов, приведен в рецепте: создается вспомогательное квадратное изображение с шириной и высотой равной радиусу закругления. После этого, на вспомогательное изображение накладывается прозрачный эллипс, который оставляет видимым только скругленный угол, все остальное перекрывается. Чтобы перекрыть часть изображения прозрачным эллипсом, необходимо выключить режим сопряжения цветов. Когда получилось создать изображение с видимым скругленным уголком и прозрачным фоном, его необходимо просто нанести на четыре угла изображения, у которого необходимо сделать закругление углов.

Пример:

```
// исходная картинка
$image = "test.jpg";
// радиус углов
$radius = 50;
// цвет, которым будет заполнен угол
$background = 0xffffff;

// загружаем картинку
```

```
$img = imagecreatefromjpeg($image);
// размер исходной картинки
$width = imagesx($img);
$height = imagesy($img);
// создаем изображение для углов
$corner = imagecreatetruecolor($radius, $radius);
// выключаем режим сопряжения цветов
imagealphablending($corner, false);
// прозрачный цвет
$trans = imagecolorallocatealpha(
  $corner, 255, 255, 255, 127
);
// заливаем картинку для углов
imagefill($corner, 0, 0, $background);
// рисуем прозрачный эллипс
imagefilledellipse(
  $corner, $radius, $radius,
  $radius * 2, $radius * 2, $trans
);
// массив положений.
// Для расположения эллипсов по углам
$positions = array(
  array(0, 0),
  array($width - $radius, 0),
  array($width - $radius, $height - $radius),
  array(0, $height - $radius),
);
// накладываем на углы картинки
// изображения с эллипсами
foreach ($positions as $pos) {
  // копируем эллипс на картинку
  imagecopyresampled(
        $img, $corner, $pos[0], $pos[1], 0, 0,
        $radius, $radius, $radius, $radius
  );
  // поворачиваем изображение с эллипсом
  // каждый раз на 90 градусов
  $corner = imagerotate(
      $corner, -90, $background, false
  );
}

// заголовок для браузера
```

```
header('Content-type: image/png');
// выводим картинку в браузере
imagepng($img);
// чистим память
imagedestroy($img);
```

23. Зеркальное отображение картинки

Для зеркального отображения картинки необходимо просто перебрать все пиксели с исходного изображения и перенести их в обратном порядке на чистый холст.

Пример:

```
// исходная картинка
$image = "test.jpg";

// загружаем картинку
$source = imagecreatefromjpeg($image);
// получаем размеры картинки
$size = getimagesize($image);
// создаем новое изображение. пустой холст
$img = imagecreatetruecolor($size[0], $size[1]);
// наносим попиксельно изображение в обратном порядке
for ($x = 0; $x < $size[0]; $x++) {
  for ($y = 0; $y < $size[1]; $y++) {
    $color = imagecolorat($source, $x,$y);
    imagesetpixel($img, $size[0]-$x, $y, $color);
  }
}

// заголовок для браузера
header('Content-type: image/png');
// выводим картинку в браузере
imagepng($img);
// чистим память
imagedestroy($img); imagedestroy($source);
```

24. Создание черно-белой картинки из цветной

Для преобразования цветной картинки в черно-белую, необходимо создать пустой холст размером, как и исходная картинка. После этого, необходимо задать пустому изображению черно-белую (серую) палитру. И этих приготовлений остается только объединить исходное изображение с пустым холстом и вывести результат.

Пример:

```
// исходная картинка
$image = "test.jpg";

// получаем размеры исходного изображения
$imgSize = getimagesize($image);
$width = $imgSize[0];
$height = $imgSize[1];
// создаем новое изображение
$img = imagecreate($width,$height);
// задаем серую палитру для нового изображения
for ($color = 0; $color <= 255; $color++) {
  imagecolorallocate($img, $color, $color, $color);
}
// создаем изображение из исходного
$source = imagecreatefromjpeg($image);
// объединяем исходное изображение и серое
imagecopymerge(
  $img, $source, 0, 0, 0, 0,
  $width, $height, 100
);

// заголовок для браузера
header('Content-type: image/png');
// выводим картинку в браузере
imagepng($img);
// чистим память
imagedestroy($img);
imagedestroy($source);
```

25. Перевод цвета из HEX в RGB

В рецепте приведен пример перевода кода цвета из HEX в RGB.

Пример:

```
// перевод цвета из HEX в RGB
function hexToRgb($color) {
    // проверяем наличие # в начале,
    // если есть, то отрезаем ее
    if ($color[0] == '#') {
        $color = substr($color, 1);
    }

    // разбираем строку на массив
    if (strlen($color) == 6) {
        // если hex цвет в полной форме - 6 символов
        list($red, $green, $blue) = array(
            $color[0] . $color[1],
            $color[2] . $color[3],
            $color[4] . $color[5]
        );
    } elseif (strlen($cvet) == 3) {
        // если hex цвет в сокращенной
        // форме - 3 символа
        list($red, $green, $blue) = array(
            $color[0]. $color[0],
            $color[1]. $color[1],
            $color[2]. $color[2]
        );
    }else{
        return false;
    }

    // переводим шестнадцатеричные числа в десятичные
    $red = hexdec($red);
    $green = hexdec($green);
    $blue = hexdec($blue);

    // вернем результат
    return array(
        'red' => $red,
```

```
        'green' => $green,
        'blue' => $blue
    );
}

// пример использования
$colorHex = '#FFAA00';
$result = hexToRgb($colorHex);
var_dump($result);
```

26. Перевод цвета из RGB в HEX

В рецепте приведен пример перевода кода цвета из RGB в HEX.

Пример:

```
// перевод цвета из RGB в HEX
function rgbToHex($color) {
    $red = dechex($color[0]);
    $green = dechex($color[1]);
    $blue = dechex($color[2]);
    return "#" . $red . $green . $blue;
}

// пример использования
$colorRgb = array(125, 255, 0);
$result = rgbToHex($colorRgb);
var_dump($result);
```

27. Генерация случайной капчи

Не редко, для защиты от спама формы, например, обратной связи, используется капча – картинка с набором символов. Как реализовать саму картинку-капчу, показано в этом рецепте.

Пример:

```php
// список символов, используемых в капче
$let = '0123456789ABCDEFGH';
// количество символов в капче
$len = 4;
// шрифт
$font = 'impact.ttf';
// Размер шрифта
$fontsize = 20;
// Размер капчи
$width = 100;
$height = 30;

// создаем изображение
$img = imagecreatetruecolor($width, $height);
// фон
$white = imagecolorallocate($img, 220, 220, 220);
imagefill($img, 0, 0, $white);
// Переменная, для хранения значения капчи
$capchaText = '';

// Заполняем изображение символами
for ($i = 0; $i < $len; $i++){
    // Из списка символов, берем случайный символ
    $capchaText .= $let[rand(0, strlen($let)-1)];
    // Вычисляем положение одного символа
    $x = ($width - 20) / $len * $i + 10;
    $y = $height - (($height - $fontsize) / 2);
    // Укажем случайный цвет для символа
    $color = imagecolorallocate(
        $img, rand(0, 150),
        rand(0, 150), rand(0, 150)
    );
    // Генерируем угол наклона символа
    $naklon = rand(-30, 30);
    // Рисуем символ
    imagettftext(
       $img, $fontsize, $naklon, $x,
       $y, $color, $font, $capchaText[$i]
    );
}

// заголовок для браузера
header('Content-type: image/png');
```

```
// вывод капчи на страницу
imagepng($img);
// чистим память
imagedestroy($img);
```

28. Генерация арифметической капчи

В предыдущем рецепты уже был рассмотрен пример генерации капчи. Но прошлый раз выводились просто символы. Эту капчу можно немного модернизировать, реализовать ее в виде арифметического примера – например, вычисление суммы.

Пример:

```
// шрифт
$font = 'impact.ttf';
// Размер шрифта
$fontsize = 20;
// Размер капчи
$width = 120;
$height = 40;

// придумываем пример для капчи
$a = mt_rand(1, 19);
$b = mt_rand(1, 19);
$capchaText = $a . '+' . $b . '=';
// Ответ на пример
$capchaResult = $a + $b;

// создаем изображение
$img = imagecreatetruecolor($width, $height);
// фон
$white = imagecolorallocate($img, 220, 220, 220);
imagefill($img, 0, 0, $white);

// Заполняем изображение символами
for ($i = 0; $i < strlen($capchaText); $i++){
    // Из списка символов, берем случайный символ
    $litteral = $capchaText[$i];
```

```
    // Вычисляем положение одного символа
    $x = ($width - 20) / strlen($capchaText) * $i +10;
    $y = $height - (($height - $fontsize) / 2);
    // Сгенерируем случайный цвет для символа.
    $color = imagecolorallocate(
        $img, rand(0, 150),
        rand(0, 150), rand(0, 150)
    );
    // Генерируем угол наклона символа
    $naklon = rand(-10, 10);
    // Рисуем один символ
    imagettftext(
        $img, $fontsize, $naklon, $x, $y,
        $color, $font, $litteral
    );
}

// Добавим на капчу несколько рандомных полосок
for ($i = 0; $i < $countLine; $i++){
    // сгенерируем координаты для линии
    $part = $width/100; // длина картинки в процентах
    // x1 не больше чем до 30% картинки
    $x1 = mt_rand(0, round($part*30));
    $y1 = mt_rand(0, $height);
    // x2 не меньше чем от 70% картики
    $x2 = mt_rand(round($part*70), round($part*100));
    $y2 = mt_rand(0, $height);
    // сгенерируем случайный цвет для линии
    $color = imagecolorallocate(
        $img, rand(0, 150),
        rand(0, 150), rand(0, 150)
    );
    imageline ($img, $x1, $y1, $x2, $y2, $color);
}

// заголовок для браузера
header('Content-type: image/png');
// вывод капчи на страницу
imagepng($img);
// чистим память
imagedestroy($img);
```

29. Вывод случайной картинки

Рецепт показывает, как можно реализовать вывод случайной картинки на странице сайта.

Пример:

Для начала необходимо вывести на странице следующий код:

```
<img src="/randomImg.php" />
```

Код скрипта **randomImg.php**:

```
// устанавливаем заголовок для вывода картинки
header("Content-type: image/jpeg");
// массив картинок
$imgPathArray = array(
   'test.jpg',
   'test2.jpg',
   'test3.jpg'
);
// берем случайную картинку
$imgPath = $imgPathArray[
   mt_rand(0, count($imgPathArray)-1)
];
// загружаем картинку
$img = imagecreatefromjpeg($imgPath);
// выводим картинку в браузере
imagejpeg($img);
// очищаем память
imageDestroy($image);
```

VIII. Работа с базой данных(MySql)

1. Подключение к базе данных

Для подключения к базе данных, сначала необходимо установить соединение с MySql, а после этого выбрать нужную базу данных. Подключение к MySql осуществляется с помощью php функции **mysql_connect**, которой необходимо передать адрес, по которому расположен MySql, а также логин и пароль пользователя. После успешного соединения с MySql, происходит выбор базы данных, это реализовывается с помощью php функции **mysql_selectdb**.

Если соединение не было установлено, то ошибки можно вывести с помощью функции **mysql_error**, которая возвращает текст ошибки последней операции с MySql.

Пример:

```
// устанавливаем константы
// обычно это прописывается в отдельном файле
// адрес, по которому расположен MySql
define("HOST", "localhost");
// имя пользователя MySql
define("USER", "root");
// пароль к MySql
define("PASSWORD", "");
// название БД
define("DB_NAME", "mydb");

//подключение к MySql
$db_connect = mysql_connect(HOST, USER, PASSWORD,
TRUE);
if (!$db_connect) {
    die('Ошибка подключения: ' . mysql_error());
}
// выбор базы данных
mysql_selectdb(DB_NAME, $db_connect);
echo 'Подключение прошло успешно';
```

2. Установка кодировки соединения с БД

Иногда возникают проблемы с отображением данных, полученных из базы данных, нередко проблема кроется в кодировке соединения. В php имеется две функции для работы с кодировкой соединения. Первая функция – **mysql_client_encoding**, проверяет текущую кодировку, а вторая – **mysql_set_charset** – устанавливает кодировку.

Пример:

```
define("HOST", "localhost");
// имя пользователя MySql
define("USER", "root");
// пароль к MySql
define("PASSWORD", "");
// название БД
define("DB_NAME", "mydb");

//подключение к MySql
$db_connect = mysql_connect(HOST, USER, PASSWORD,
TRUE);
if (!$db_connect) {
    die('Ошибка подключения: ' . mysql_error());
}
// выбор базы данных
mysql_selectdb(DB_NAME, $db_connect);

// проверяем кодировку
$charset = mysql_client_encoding($db_connect);
echo $charset;

// устанавливаем кодировку
mysql_set_charset("utf8");

// еще раз проверяем кодировку
$charset = mysql_client_encoding($db_connect);
echo $charset;
```

3. Запись данных в таблицу БД. INSERT

Чтобы реализовать запись в таблицу можно воспользоваться php функцией **mysql_query**, которая посылает SQL-запросы к базе данных, с которой установлено соединение. В случае успешного выполнения запроса **mysql_query** вернет значение true, а при ошибке – false.

Для создания записи в таблице базы данных SQL-запрос будет выглядеть следующим образом:

```
INSERT INTO `tbl_users` (`first_name`, `last_name`)
VALUES ('Иван', 'Иванов');
```

Где `tbl_users` – название таблицы, `first_name` и `last_name` – имена заполняемых полей, а «Иван» и «Иванов» - это значения, которые будут записаны в соответствующие поля таблицы.

Пример:

```
define("HOST", "localhost");
// имя пользователя MySql
define("USER", "root");
// пароль к MySql
define("PASSWORD", "");
// название БД
define("DB_NAME", "mydb");

//подключение к MySql
$db_connect = mysql_connect(HOST, USER, PASSWORD,
TRUE);
if (!$db_connect) {
    die('Ошибка подключения: ' . mysql_error());
}
// выбор базы данных
mysql_selectdb(DB_NAME, $db_connect);
```

```
// устанавливаем кодировку
mysql_set_charset("utf8");

// данные для записи
$first_name = "Иван";
$last_name = "Иванов";
// создание записи в таблице
$result = mysql_query("
   INSERT INTO `tbl_users` (`first_name`, `last_name`)
         VALUES ('{$first_name}', '{$last_name}')
");
if($result){
   echo "Запись успешно создана!";
}else{
   echo "Ошибка: запись не создана!";
}
```

4. Получение ID последней записи

После успешного выполнения запроса на запись данных в таблицу базы данных, с помощью функции **mysql_insert_id**, можно получить идентификатор (ID) только что созданной записи.

Пример:

```
define("HOST", "localhost");
define("USER", "root");
define("PASSWORD", "");
define("DB_NAME", "test_db");

//подключение к MySql
$db_connect = mysql_connect(HOST, USER, PASSWORD,
TRUE);
if (!$db_connect) {
    die('Ошибка подключения: ' . mysql_error());
}
// выбор базы данных
mysql_selectdb(DB_NAME, $db_connect);
// устанавливаем кодировку
```

```
mysql_set_charset("utf8");

$first_name = "Иван";
$last_name = "Иванов";
// создание записи в таблице
$result = mysql_query("
   INSERT INTO `tbl_users` (`first_name`, `last_name`)
         VALUES ('{$first_name}', '{$last_name}')
");

// получение ID последней записи
$insert_id = mysql_insert_id();
echo "ID записи: $insert_id";
```

5. Экранирование данных перед записью в БД

В предыдущих рецептах происходила запись данных в таблицу БД в таком виде, каком они есть. Так делать можно только в том случае, если данные генерируются на сервере. Но если нужно записать данные, которые отчасти или полностью пришли с клиента, например, по средствам POST или GET запроса, то осуществлять запись без экранирования данных нельзя. Поскольку это может привести к созданию «дыры» в коде, через которую будет очень просто взломать или навредить работе базе данных или всего сайта.

Экранировать данные можно несколькими способами, если данные имеют числовое значение, то их нужно привести к числовому виду – целому числу или числу с плавающей точки, с помощью функций **`int`** или **`float`**. А если данные имеют строковый тип, то экранировать их нужно функцией **`mysql_real_escape_string`**.

Пример:

```
define("HOST", "localhost");
define("USER", "root");
define("PASSWORD", "");
```

```php
define("DB_NAME", "test_db");

//подключение к MySql
$db_connect = mysql_connect(HOST, USER, PASSWORD,
TRUE);
if (!$db_connect) {
    die('Ошибка подключения: ' . mysql_error());
}
// выбор базы данных
mysql_selectdb(DB_NAME, $db_connect);
// устанавливаем кодировку
mysql_set_charset("utf8");

// получаем данные из POST запроса
// имя
$first_name = $_POST['first_name'];
// фамилия
$last_name = $_POST['last_name'];
// возраст
$age = $_POST['age'];

// экранируем данные
// строки
$first_name = mysql_real_escape_string($first_name);
$last_name = mysql_real_escape_string($last_name);
// число
$age = (int)$age;

// создание записи в таблице
$result = mysql_query("
  INSERT INTO `tbl_users` (`f_name`, `l_name`, `age`)
  VALUES ('{$first_name}', '{$last_name}', {$age})
");
```

6. Получение данных из БД. SELECT

Для получения данных из таблицы, можно воспользоваться ранее описанной функцией **mysql_query**, которая отправляет SQL-запросы к базе данных.

Чтобы получить данные, запрос должен выглядеть следующим образом: `SELECT * FROM tbl_users`, где `tbl_users` – название таблицы, а * - поля таблицы, значения которых необходимо получить. *(звездочка), обозначает, что будут получены все поля, если необходимо получить только некоторые, то их необходимо указать через запятую.

После выполнения запроса, необходимо обработать полученные данные, это можно реализовать с помощью нескольких php функций:

mysql_fetch_array – обрабатывает запрос и возвращает ассоциативный или числовой массив с данными. Для определения типа массива необходимо передавать флаг MYSQL_ASSOC (ассоциативный массив) или MYSQL_NUM(числовой массив), также есть третий флаг, который установлен по умолчанию MYSQL_BOTH(данные будут иметь как ассоциативные, так и числовые ключи).

mysql_fetch_row – обрабатывает запрос и возвращает массив данных с числовыми индексами.
mysql_fetch_assoc – обрабатывает запрос и возвращает массив данных с ассоциативными индексами(именами столбцов).
mysql_fetch_object – обрабатывает запрос и возвращает объект содержащий данные.

Все эти функции работают одинаково, отличие только в виде данных, которые эти функции возвращают. Все, кроме первой, функции принимают по одному параметру – обрабатываемый результат запроса.

Пример:

```
// предварительно, необходимо реализовать
// подключение к базе данных
```

```php
// запрос на получение данных
// при ошибке остановим скрипт и выведем ошибку
$query = mysql_query("
   SELECT * FROM tbl_users
") or die(mysql_error());

// массив для данных
$users = array();
// перебираем все полученные данные по одной строке
// и записываем их в массив
while($row = mysql_fetch_array($query, MYSQL_ASSOC)){
   $users[] = $row;
}
// вывод результата
print_r($users);
```

7. Получение данных с фильтрацией. WHERE

В прошлом рецепте было реализовано получение всех записей из таблицы. Такие запросы в реальном мире встречаются крайне редко, поскольку практически всегда необходимо получить данные соответствующие определенному условию, например, всех пользователей с фамилией «Иванов». Чтобы добавить условие в SQL-запрос, нужно дописать в него оператор WHERE:

```
SELECT * FROM `tbl_users`
    WHERE `last_name` = "Иванов"
```

Помимо жесткого сравнения, можно реализовать условие, при котором выборка будет реализована по тем пользователям, чья фамилия оканчивается на «ов». Для этого вместо знака «=» необходимо задать оператор LIKE. Он осуществляет поиск по маске, которая собирается из различных модификаторов, самый распространённый из них «%» - обозначает, что на его месте может

быть любое количество любых символов. Таким образом, запрос будет выглядеть вот так:

```
SELECT * FROM `tbl_users`
  WHERE `last_name` LIKE "%ов"
```

Пример:

```
// предварительно, необходимо реализовать
// подключение к базе данных

// запрос
// при ошибке остановим скрипт и выведем ошибку
$query = mysql_query("
  SELECT * FROM `tbl_users`
        WHERE `last_name` LIKE '%ов'

") or die(mysql_error());

// получение резальтата запроса
$result = array();
while($row = mysql_fetch_array($query, MYSQL_ASSOC)){
  $result[] = $row;
}
// вывод результата
print_r ($result);
```

8. Количество записей(строк) в результате

При необходимости, сразу после выполнения SQL-запроса, можно узнать количество рядов результата, без предварительной обработки запроса. Для этого можно использовать php функцию **mysql_num_rows**.

Пример:

```
// предварительно, необходимо реализовать
// подключение к базе данных
```

```php
// запрос на получение данных
// при ошибке остановим скрипт и выведем ошибку
$query = mysql_query("
  SELECT * FROM tbl_users
") or die(mysql_error());

// получение количества строк
$count_row = mysql_num_rows($query);
echo "Строк: $count_row";
```

9. Обновить запись в таблице. UPDATE

Обновить запись в таблице базы данных можно с помощью отправки SQL-запроса, функцией **mysql_query**.

Пример SQL-запроса:

```sql
UPDATE `tbl_users` SET `last_name` = 'Сидоров'
  WHERE `id` = 3
```

Где `tbl_users` – имя таблицы, запись в которой необходимо обновить. `last_name` – поле, в которое будет записано новое значение - `'Сидоров'`. **WHERE** `` `id` = 3 `` – это условие, по которому будет найдена нужная запись.

Пример:

```php
// предварительно, необходимо реализовать
// подключение к базе данных

// запрос на обновление данных
// при ошибке остановим скрипт и выведем ошибку
mysql_query("
  UPDATE `tbl_users` SET `last_name` = 'Сидоров'
    WHERE `id` = 5
") or die(mysql_error());
```

10. Удаление записи в таблице. DELETE

Для удаления записи, как и в предыдущих рецептах, для каких-либо взаимодействий с базой данных и ее таблицами, необходимо отправлять SQL-запрос с помощью функции **mysql_query**.

Запрос на удаление записи выглядит не сложнее других:

```
DELETE FROM `tbl_users` WHERE `id` = 3
```

В запросе `tbl_users` означает имя таблицы, в которой будет удалена запись, а **WHERE** `id` = 3 является условием, для выбора конкретной записи.

Пример:

```
// предварительно, необходимо реализовать
// подключение к базе данных

// запрос на удаление данных
// при ошибке остановим скрипт и выведем ошибку
mysql_query("
   DELETE FROM `tbl_users` WHERE `id` = 3
") or die(mysql_error());
```

11. Получение данных с лимитом строк. LIMIT

Получить данные по определенному лимиту, например, не больше трех строк, можно с помощью немного модифицированного SQL-запроса для получения данных из таблицы, который был описан в рецепте «Получение данных из БД». SQL-запрос должен быть дополнен оператором **LIMIT**, выглядеть новый запрос будет так:

```
SELECT * FROM `tbl_users` LIMIT 5, 10
```

Оператор **LIMIT** может принимать два числовых параметра, один из них не обязательный. Если задан только один параметр, то он будет ограничивать количества получаемых записей из таблицы. В случае, когда заданы два параметра, как в примере, то первое число будет обозначать номер записи, с которой начнется выборка, а второе число – это максимальное количество записей, которые будут получены.

Пример:

```
// предварительно, необходимо реализовать
// подключение к базе данных

// запрос
// при ошибке остановим скрипт и выведем ошибку
$query = mysql_query("
  SELECT * FROM `tbl_users` LIMIT 5, 10
") or die(mysql_error());

// массив для данных
$users = array();
// перебираем все полученные данные по одной строке
// и записываем их в массив
while($row = mysql_fetch_array($query, MYSQL_ASSOC)){
  $users[] = $row;
}
// вывод результата
print_r($users);
```

12. Получение отсортированных данных. ORDER BY

При SQL-запросе на получение данных, можно сразу реализовать сортировку строк по необходимому столбцу или по нескольким столбцам. Для сортировки данных необходимо дописать оператор

ORDER BY в конец SQL-запроса, осуществляющего выборку данных. Запрос будет выглядеть следующим образом:

```
SELECT * FROM `tbl_users` ORDER BY `name` DESC
```

При использовании оператора **ORDER BY**, необходимо указать название столбца, по которому будет произведена сортировка, в приведенном примере - это `tbl_users`. Также можно задать необязательный флаг, который определяет направление сортировки: **DESC** – от большего к меньшему, или **ASC** – от меньшего к большему.

Пример:

```
// предварительно, необходимо реализовать
// подключение к базе данных

// запрос
// при ошибке остановим скрипт и выведем ошибку
$query = mysql_query("
   SELECT * FROM `tbl_users` ORDER BY `first_name` DESC
") or die(mysql_error());

// массив для данных
$users = array();
// перебираем все полученные данные по одной строке
// и записываем их в массив
while($row = mysql_fetch_array($query, MYSQL_ASSOC)){
   $users[] = $row;
}
// вывод результата
print_r($users);
```

13. Наибольшее и наименьшее значение в таблице

Получить максимальное и минимальное значение из таблицы можно, добавив в SQL-запрос две специальные функции – **max** и **min**, которые осуществляют выборку:

```
SELECT
   max(`age`) as max_age,
   min(`age`) as min_age
    FROM `tbl_users`
```

В примере происходит поиск наибольшего и наименьшего возраста в таблице `tbl_users`. Результатом такого запроса будет одна строка с двумя значениями – `max_age` и `min_age`.

Пример:

```
// предварительно, необходимо реализовать
// подключение к базе данных

// запрос
// при ошибке остановим скрипт и выведем ошибку
$query = mysql_query("
   SELECT
         max(`age`) as max_age,
         min(`age`) as min_age
   FROM `tbl_users`
") or die(mysql_error());

$row = mysql_fetch_array($query, MYSQL_ASSOC);
// вывод результата
echo "Максимальный возраст: {$row['max_age']}<br/>";
echo "Минимальный возраст: {$row['min_age']}";
```

14. Получение сгруппированных строк. GROUP BY

Иногда необходимо получить какие-то данные для сгруппированных строк, например, узнать максимальный возраст для всех Ивановых, Петровых и тд. То есть необходимо

сгруппировать всех пользователей по фамилии и получить для них максимальный возраст. Для этих целей можно воспользоваться SQL-запросом с оператором **GROUP BY**. Запрос будет выглядеть так:

```
SELECT `last_name`, max(`age`) as max_age
FROM `tbl_users`
  GROUP BY `last_name`
```

Результатом выполнения такого запроса будет массив строк, которые будет содержать по два значения – фамилию и максимальный возраст для группы этих фамилий.

Пример:

```
// предварительно, необходимо реализовать
// подключение к базе данных

// запрос
// при ошибке остановим скрипт и выведем ошибку
$query = mysql_query("
  SELECT last_name, max(`age`) as max_age
  FROM `tbl_users`
       GROUP BY last_name
") or die(mysql_error());

// получение результата запроса
$result = array();
while($row = mysql_fetch_array($query, MYSQL_ASSOC)){
  $result[] = $row;
}

// вывод результата
print_r ($result);
```

15. Выборка по сгруппированным строкам. HAVING

В предыдущем рецепте, с помощью **GROUP BY** были сгруппированы строки - группировались пользователи по фамилиям и для каждой группы находился самый большой возраст. Иногда бывает необходимо помимо всего этого, еще и отсортировать полученные группы по какому-нибудь условию. Например, получить группы фамилий, у которых минимальный возраст превышает 18 лет. В таких случаях можно воспользоваться оператором **HAVING**, он добавляется в SQL-запрос после группировки:

```
SELECT last_name, min(`age`) as min_age
FROM `tbl_users`
  GROUP BY last_name
  HAVING min_age > 18
```

Пример:

```
// предварительно, необходимо реализовать
// подключение к базе данных

// запрос
// при ошибке остановим скрипт и выведем ошибку
$query = mysql_query("
  SELECT last_name, min(`age`) as min_age
  FROM `tbl_users`
       GROUP BY last_name
       HAVING min_age > 18

") or die(mysql_error());

// получение результата запроса
$result = array();
while($row = mysql_fetch_array($query, MYSQL_ASSOC)){
  $result[] = $row;
}

// вывод результата
print_r ($result);
```

16. Получение данных из нескольких таблиц. JOIN

Для получения записей сразу из нескольких таблиц можно воспользоваться оператором **JOIN**, который объединяет таблицы. Например, необходимо получить все заказы пользователя. Пользователи находятся в одной таблице(`tbl_users`), а заказы в другой(`tbl_orders`). В таком случае, необходимо связать эти таблицы, по полю ID пользователя, которое должно быть в таблице пользователей и в таблице с заказами. Для получения заказов пользователя с ID 1, запрос будет выглядеть следующим образом:

```
SELECT * FROM `tbl_users`
  JOIN `tbl_orders`
       ON  `tbl_orders`.`id_user` = `tbl_users`.`id`
  WHERE `tbl_users`.`id` = 1
```

В операторе **JOIN** должна быть указана таблица, которая присоединяется, а так же условие, по которому будет реализована связь таблиц, в данном случае это id пользователя.

Если для пользователя не будет найдено ни одного заказа, то результат выполнения запроса будет пустым. Если необходимо получить данные о пользователе независимо от существования заказов, то перед оператором **JOIN** необходимо дописать **LEFT**, в таком случае запрос будет таким:

```
SELECT * FROM `tbl_users`
  LEFT JOIN `tbl_orders`
       ON  `tbl_orders`.`id_user` = `tbl_users`.`id`
  WHERE `tbl_users`.`id` = 1
```

Пример:

```
// предварительно, необходимо реализовать
// подключение к базе данных

// запрос
```

```php
$query = mysql_query("
   SELECT * FROM `tbl_users`
     LEFT JOIN `tbl_orders`
       ON  `tbl_orders`.`id_user` = `tbl_users`.`id`
     WHERE `tbl_users`.`id` = 1
") or die(mysql_error());

// получение результата запроса
$result = array();
while($row = mysql_fetch_array($query, MYSQL_ASSOC)){
   $result[] = $row;
}

// вывод результата
print_r ($result);
```