Содержание

ВВЕДЕНИЕ

В данной лабораторной работе Вы познакомитесь с синтаксисом и основными конструкциями языка PHP. Получите навыки взаимодействия с СУБД MySQL, а так же с сессиями и cookie, научитесь создавать динамические изображения, используя средства языка PHP. Данные знания позволят Вам реализовывать простые веб-сайты. В ходе лабораторной работе будет рассмотрен пример разработки гостевой книги с разграничением прав доступа и возможностью предотвращения множественных отправок сообщений специализированными скриптами (роботами), а также возможностью загрузки изображения на сервер.

В следующей лабораторной работе, мы модифицируем гостевую книгу, добавив возможность оставлять/редактировать/удалять записи пользователями без перезагрузки страницы. Для реализации данной возможности мы будем использовать язык JavaScript с использованием AJAX и формата данных JSON.

1 Подготовка к выполнению лабораторной работе

Как вам известно, для разработки и отладки веб-сайта необходим веб-сервер с предустановленным интерпретатором скриптового языка PHP. В первой лабораторной работе мы устанавливали свой собственный веб-сервер на виртуальной машине, вы можете смело его использовать, загружая файлы на виртуальную машину, используя ftp. Этот вариант не совсем удобный, так как нужно будет постоянно загружать модифицированные файлы на веб-сервер, что займет некоторое время. Поэтому для удобства работы советую установить локальный веб-сервер OpenServer, дистрибутив и инструкцию по установке вы можете найти на сетевом компьютере \\112b-03. Но для разработки веб-сайта не достаточно установки одного лишь веб-сервера, необходимо установить и среду разработки. Существует огромное количество замечательных сред разработки веб-приложений таких как: PHPStorm, Zend Studio, Eclipse, NetBeans и д.р. Вы можете выбрать удобную для Вас среду разработки, я совету использовать Eclipse потому что это не только среда разработки, это платформа, которую можно расширить модулями и использовать для разработки практически на любом языке программирования. Инструкцию по установке Eclipse с модулем для разработки с использованием языка PHP и настройка взаимодействия с ftp-сервером вы можете найти на сервере \\112b-03.

2 Основы языка сценариев PHP

Во второй лабораторной работе вы научились создавать статические веб-страницы с использованием языка разметки HTML и CSS, но как вы понимаете, редко на каких веб-сайтах используются статические страницы, потому что для редактирования содержимого страницы необходимо модифицировать её исходный код. Для создания динамических страниц можно использовать различные языки программирования: PHP, Perl, Ruby (Ruby on Rails) и другие. Но в рамках данной лабораторной работы мы рассмотрим язык PHP, так как он является одним из самых распространенных и простых в изучении.

Аббревиатура PHP означает «Hypertext Preprocessor» (Препроцессор Гипертекста), если понимать расшифровку аббревиатуры дословно, то с использованием языка PHP осуществляется формирование гипертекстового документа.

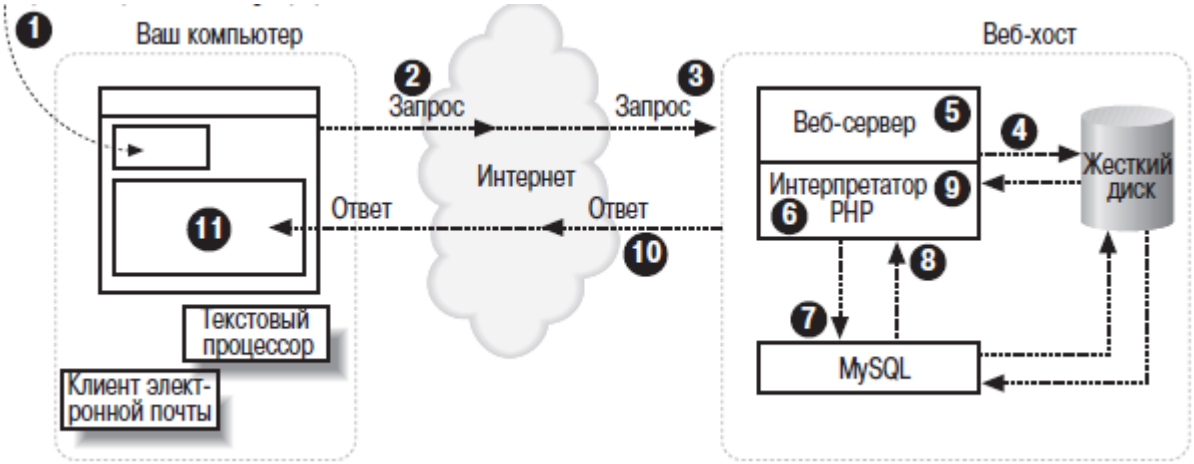

Рисунок 1 – Совместные действия интерпретатора PHP, MySQL и веб-сервера по созданию страницы

На рисунке 1 приведена последовательность действий, выполняемых каждый раз, когда клиент запрашивает веб-страницу содержащую PHP-код, который обращается к БД для извлечения информации.

1. клиент указывает адрес веб-страницы (например, http://it.aics.ru/login.php) в адресной строке браузера;
2. браузер разбивает адрес на составляющие и отправляет имя страницы веб-серверу;
3. процесс веб-сервера принимает запрос на получение страницы login.php;
4. веб-сервер считывает файл login.php с жесткого диска хоста;
5. веб-сервер определяет, что это сценарий PHP, а не простой HTML-файл, и поэтому передает его на обработку интерпретатору PHP;
6. интерпретатор PHP исполняет PHP-код, который он обнаружил в тексте, полученном от процесса веб-сервера. Этот код включает в себя обращения к базе данных MySQL;
7. интерпретатор PHP запрашивает у процесса базы данных MySQL обработку обращений к базе данных;
8. процесс базы данных MySQL возвращает результаты запросы к базе данных;
9. интерпретатор PHP завершает исполнение PHP-кода, добавляя данные, полученные из базы данных, и возвращает результат процесса веб-сервера;
10. веб-сервер возвращает результат браузера в виде HTML-текста;
11. веб-браузер формирует внешний вид веб-страницы на экране вашего компьютера в соответствии с полученным HTML-текстом.

Чтобы добавить php-код на страницу его необходимо поместить между тегами <?php и ?>, а расширение файла с php-кодом должно быть *.php.

```
<!DOCTYPE html>
<html lang="ru">
    <head>
        <meta charset="utf-8" />
        <title>
            Название страницы
        </title>
     </head>
     <body>
         <?php
              //PHP-код
         ?>
     </body>
</html>
```

Все что вы поместите между открывающим тегом <?php и закрывающим ?> будет интерпретироваться веб-сервером как PHP, что в свою очередь означает – веб-сервер будет обрабатывать этот код. Весь же остальной код, который находится за пределами тегов PHP будет немедленно отправляться в веб-браузер как обычный HTML.

2.1 Комментирование кода

Одним из важных шагов в процессе разработки является документирования программного кода, в языке PHP существует три способа комментирования.

Однострочные комментарии:

```
# Это комментарий
// Это тоже комментарий
```

Многострочный комментарий:

```
/* Это многострочный комментарий.
   Его можно использовать для комментирования
   двух и более строчек текста */
```

2.2 Переменные

Переменные PHP являются контейнером для хранения данных. Переменные состоят из двух частей: идентификатора (имени) и значения. Идентификатор представляет собой последовательность букв, цифр, символов подчеркивания без пробелов и знаков препинания. Идентификатор обязательно начинается с символа доллара($), за которым должна следовать буква или символ подчеркивания. Значением переменной являются данные, которые содержатся в переменной. Данные хранящиеся в переменных могут быть

определенных типов, PHP автоматически выбирает тип переменной, соответствующий присвоенному значению.

$_идентификатор=значение;

```
<?php
    $name = 'John';
?>
```

Работая с переменными, вы можете совершать над ними следующие действия: получать значения переменных, присваивать значения, проверять на существования, удалять, а также явно приводить переменные к определенному типу.

2.2.1 Вывод значения переменной на экран

Значения переменных можно вывести на экран с помощью оператора echo или функции print. В интернете есть много статей, какой из способов вывода выбрать. Лучшим вариантом является использование оператора echo с несколькими параметрами, дает весьма ощутимый выигрыш в производительности, для примера:

```
<?php print $value . '<br />'; ?> - 1,462 c
<?php echo $value, '<br />'; ?> - 1,321 c
```

Для вывода более подробной информации о переменных, которая может понадобиться при отладке программы, служат функции var_dump, print_r.

2.2.2 Проверка существования переменных PHP

Проверка существование переменных осуществляется при помощи оператора isset(имя_переменной1, имя_переменной2, ... , имя_переменной_N). Данная функция возвращает true если переменная существует и false в противном случае. Пример:

```
<?php
if(isset($myvar))
    echo 'Такая переменная есть. Ее значение', $myvar;
?>
```

Важно знать, что вы не можете использовать неинициализированную переменную в программе – иначе при выполнении сценария интерпретатором, возникнет предупреждение.

2.2.3 Удаление переменных

Удаление переменной PHP осуществляется при помощи оператора – unset(имя_переменной). Использование unset для работы с обычными переменными редко

бывает целесообразно. Чаще всего данный оператор используется для удаления элемента в массиве.

2.2.4 Определение/установка типа переменной

Существуют еще несколько стандартных функций, которые занимаются определением типа переменных и часто включаются в условные операторы. Вот они.

- is_integer(имя_переменной) – возвращает true, если переменная – целое число.
- is_double(имя_переменной) – возвращает true, если переменная – действительное число.
- is_string(имя_переменной) – возвращает true, если переменная является строкой.
- is_array(имя_переменной) – возвращает true, если переменная является массивом.
- is_object(имя_переменной) – возвращает true, если переменная объявлена как объект.
- is_boolean(имя_переменной) – возвращает true, если переменная определена как логическая переменная.
- gettype(имя_переменной) – возвращает строки, соответственно, со значениями: array, object, integer, double, string, boolean или unknown type в зависимости от типа переменной.

Существует функция, которая пытается привести тип указанной переменной к одному из стандартных (например, вам может понадобиться перевести строку в целое число) – settype($a,$type). Функция пытается привести тип переменной $a к типу $type. Если это сделать не удалось, возвращает false.

2.2.5 Область видимости

Область видимости переменной - это контекст, в котором эта переменная определена. В большинстве случаев все переменные PHP имеют только одну область видимости. Эта единая область видимости охватывает также включаемые (include) и требуемые (require) файлы.

```
<?php
    $a = 1; /* глобальная область видимости */
    function test() {
        echo $a; /* ссылка на переменную локальной области видимости */
    }
    test();
?>
```

В результате выполнения данного скрипта на экран ничего выведено не будет, поскольку выражение echo указывает на локальную версию переменной $a, а в пределах

этой области видимости ей не было присвоено значение. Среда Eclipse за Вас подумает и покажет следующее предупреждение.

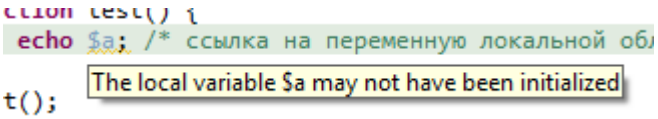

Рисунок 2 – Предупреждение в Eclipse

Глобальные переменные

Глобальные переменные позволяют вам пересекать границы между функциями, чтобы обращаться к значениям переменных. Оператор global указывает, что данная переменная будет той же самой переменной повсюду в программе, то есть глобальной переменной.

```
<?php
    $a = 1; /* глобальная область видимости */
    function test() {
        global $a;
        echo $a; /* ссылка на переменную глобальной области видимости */
    }
    test();
?>
```

Глобальные переменные следует использовать в редких случаях, поскольку легко изменить значение переменной по ошибке, не предусмотрев последствий. Ошибки такого типа бывает очень сложно обнаружить.

Статические переменные

Статические переменные – это переменные, которые не исчезают после завершения функции. Значение статической переменной можно снова использовать при следующем вызове функции – она по-прежнему будет иметь то же значение, которое получила при последнем вызове функции.

```
<?php
    $a = 10; /* глобальная область видимости */
    function test() {
        static $a = 0;
        $a = $a + 1;
        echo $a, '</br>'; /* ссылка на статическую переменную локальной области
видимости */
    }
    test();
    test();
    echo $a;
?>
```

Суперглобальные переменные

Для предоставления информации об окружении, в котором работает PHP-сценарий, PHP использует специальные переменные, которые называются суперглобальными (super

globals). Эти переменные не нужно объявлять как глобальные, они автоматически становятся общедоступными и содержат важные сведения об окружении сценария, например данные, полученные от пользователя. Начиная с версии PHP 4.0.1 суперглобальные переменные определены как массивы. Массивы – это специальные наборы значений.

Имя переменной массива	Описание
$GLOBALS	Содержит все глобальные переменные, доступные локальному сценарию. Имена переменных используются как индексы массива
$_SERVER	Содержит информацию об окружении веб-сервера
$_GET	Содержит информацию о запросах GET (при отправке форм). Эти значения следует обязательно проверять перед использованием.
$_POST	Содержит информацию о запросах POST (другой тип отправки форм). Эти значения следует обязательно проверять перед использованием
$_COOKIE	Содержит информацию о cookies HTTP
$_FILES	Содержит информацию о файлах, загружаемых методом POST
$_ENV	Содержит информацию об окружении сценариев
$_SESSION	Содержит информацию из всех переменных, зарегистрированных в рамках сессии

Примером суперглобальной переменной может служить $_SERVER["PHP_SELF"]. Эта переменная содержит имя исполняемого сценария и входит в состав массива $_SERVER.

2.2.6 Константы

Значение константы, как следует из ее названия, не может изменяться во время исполнения программы. Константы определяют с помощью функции define(), которой в первом аргументе передается имя константы, а во втором – ее значение. Константы имеют глобальную область видимости и могут принимать значение любого элементарного (скалярного) типа данных, например строки или числа. Чтобы получить значение константы, достаточно просто обратиться к ее имени или воспользоваться функцией constant. В отличие от переменных, перед именем константы знак доллара ($) не ставится.

Если имя константы хранится в переменной или возвращается функцией, то чтобы получить значение константы, необходимо воспользоваться функцией constant(имя_константы). Эта функция получает имя константы в качестве аргумента и возвращает ее значение. Кроме того, с помощью функции get_defined_constants() можно получить список (в виде массива) всех определенных вами констант.

```php
<?php
    define("HELLO", "Привет, МИР!");
    echo HELLO; // Здесь выводится "Привет, МИР!"
    $constant_name = "HELLO";
    echo constant($constant_name);
?>
```

2.3 Условные операторы

Условные операторы, как и переменные, являются стандартными блоками программ на PHP. Они изменяют поведение сценария в соответствии с заданными критериями. В языке PHP есть три первичных условных оператора:

– if;

– ? : (сокращенная форма записи оператора if);

– switch.

Операторы сравнения проверяют отношение двух значений. К ним относятся операторы меньше (<), меньше или равно (<=), больше (>) и больше или равно (>=). С помощью операторов сравнения часто проверяют, что происходит в некоторой точке программы. Логические операторы работают с логическими величинами – результатами выполнения операторов отношения, позволяя строить более сложные логические выражения; всего есть четыре логических оператора:

Чтобы проверить, имеют ли оба операнда значение TRUE, следует использовать оператор AND, который можно записать и как двойной амперсанд (&&). Оба оператора, AND и &&, являются логическими, их единственное отличие в том, что оператор && имеет более высокий приоритет, чем оператор AND. То же самое относится к операторам OR и ||. Оператор AND возвращает TRUE, только если оба операнда имеют значение TRUE; в противном случае возвращается значение FALSE.

Чтобы проверить, имеет ли хотя бы один операнд значение TRUE, следует использовать оператор OR, который можно записать и как двойную вертикальную линию (||). Этот оператор возвращает TRUE, если хотя бы один из операндов имеет значение TRUE.

Проверить, имеет ли значение TRUE только один из операндов (не оба сразу), позволяет оператор XOR. Этот оператор возвращает значение TRUE, если один и только

один из операндов имеет значение TRUE. Если оба операнда имеют значение TRUE, оператор вернет значение FALSE.

Инвертировать логическое значение можно с помощью оператора NOT, который часто записывается и в виде восклицательного знака (!). Он возвращает значение TRUE, если операнд имеет значение FALSE, и значение FALSE, если операнд имеет значение TRUE.

Инструкция if

Инструкция if позволяет исполнить блок кода, если условное выражение в этой инструкции имеет значение TRUE; в противном случае блок кода не исполняется. В качестве условия может применяться любое выражение, включающее проверки на ненулевое значение, равенство, NULL с участием переменных и значений, возвращаемых функциями.

Не важно, какие отдельные условные выражения составляют условное предложение. Если условие истинно, исполняется программный код, заключенный в фигурные скобки ({}). В противном случае PHP игнорирует его и переходит к проверке второго условия, проверяя все условные предложения, которые вы записали, пока не наткнется на инструкцию else, после чего автоматически выполнит этот блок. Инструкция else не является обязательной.

```php
<?php
    if ($username == "Admin"){
        echo('Добро пожаловать на страницу администратора.');
    } else {
        echo('Добро пожаловать на страницу пользователя.');
    }
?>
```

Оператор ?

Оператор ? – это тернарный (трехместный) оператор, который принимает три операнда. Он работает аналогично инструкции if, но возвращает значение одного из двух выражений. Выражение, которое будет вычисляться, определяется условным выражением. Двоеточие (:) служит разделителем выражений:

{выражение} ? вычислить_если_выражение_истинно :

вычислить_если_выражение_ложно;

```php
<?php
    $logged_in = TRUE;
    $user = "Admin";
    $banner = ($logged_in==TRUE)?"С возвращением, $user!":"Зарегистрируйтесь!";
    echo $banner;
?>
```

Инструкция switch

Инструкция switch сравнивает выражение с несколькими значениями. Как правило, в качестве выражения используется переменная, в зависимости от значения которой должен быть исполнен тот или иной блок кода.

```
<?php
    switch ($action) {
        case "ADD":
            echo "Выполнить добавление.";
            echo "Количество инструкций в каждом блоке не ограничено.";
        break;
        case "MODIFY":
            echo "Выполнить изменение.";
        break;
        case "DELETE":
            echo "Выполнить удаление.";
        break;
        default:
            echo "Ошибка: Допустимы только действия ADD, MODIFY и DELETE.";
    }
?>
```

2.4 Циклы

На втором месте по частоте использования, после конструкций условий (условных операторов), находятся циклы.

Циклы позволяют повторять определенное (и даже неопределенное - когда работа цикла зависит от условия) количество раз различные операторы. Данные операторы называются телом цикла. Проход цикла называется итерацией. PHP поддерживает три вида циклов:

- цикл с предусловием (while);
- цикл с постусловием (do-while);
- цикл со счетчиком (for);
- специальный цикл перебора массивов (foreach).

При использовании циклов есть возможность использования операторов break и continue. Первый из них прерывает работу всего цикла, а второй - только текущей итерации.

2.4.1 Цикл с предусловием (while)

Цикл с предусловием while работает по следующим принципам:

- Вычисляется значение логического выражения.
- Если значение истинно, выполняется тело цикла, в противном случае - переходим на следующий за циклом оператор.

Синтаксис цикла с предусловием:

```
while (логическое_выражение) {
```

```
    программный код цикла - инструкции;
}
```

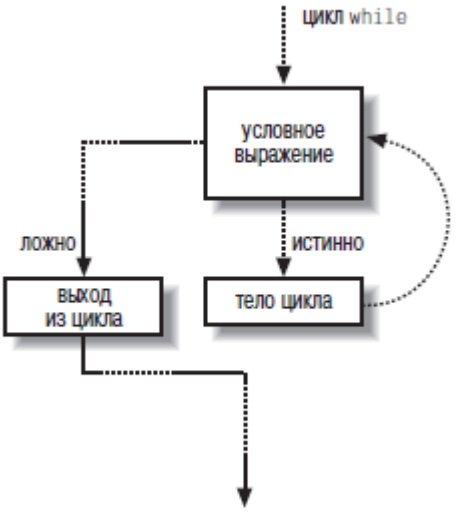

Рисунок 3 – Использование цикла while

В данном случае телом цикла является инструкция. Обычно тело цикла состоит из большого числа операторов. Приведем пример цикла с предусловием while:

```
<?php
    $num = 1;
    while ($num <= 10) {
        print "Цикл номер $num<br />\n";
        $num++;
    }
?>
```

Можно группировать операторы внутри тела цикла while, используя следующий альтернативный синтаксис:

```
while (логическое_выражение):
   программный код цикла - инструкции;
endwhile;
```

2.4.2 Цикл с постусловием (do-while)

В отличие от цикла while, этот цикл проверяет значение выражения не до, а после каждого прохода (итерации). Таким образом, тело цикла выполняется хотя бы один раз. Синтаксис цикла с постусловием такой:

```
do {
   программный код цикла - инструкции;
}
while (логическое_выражение);
```

После очередной итерации проверяется, истинно ли логическое_выражение, и, если это так, управление передается вновь на начало цикла, в противном случае цикл обрывается.

```
<?php
$num = 1;
do {
    echo " Цикл номер ".$num."<br />";
    $num++;
} while ($num <= 10);
?>
```

Циклы do ... while всегда исполняются, по крайней мере один раз. Когда исполнение дойдет до инструкции while, условное выражение вернет значение FALSE, и работа цикла do ... while прекратится.

2.4.3 Цикл со счетчиком (for)

Циклы for в общем виде предоставляют те же функциональные возможности, что и циклы while, а кроме того позволяют инициализировать и изменять значение счетчика цикла. Этот цикл имеет следующий синтаксис:

for (выражение_инициализации; условное_выражение; изменяющее_выражение){

программный код цикла;

}

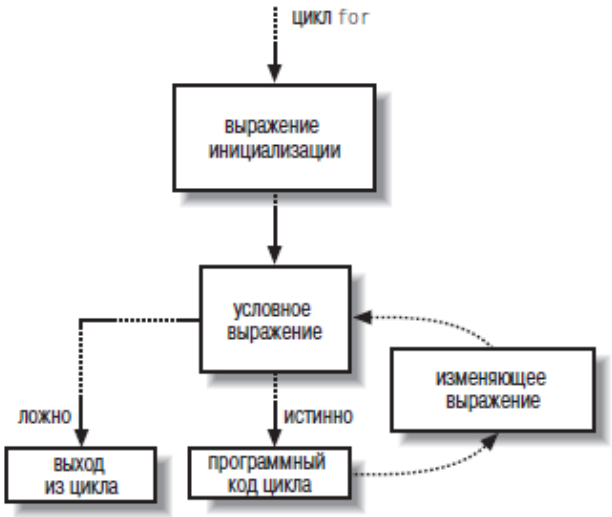

Рисунок 4 – Использование цикла for

Исполняя цикл for, PHP сначала вычисляет выражение инициализации. В каждой итерации исполняется часть кода, которая увеличивает счетчик, и затем проверяется условное выражение, чтобы узнать, не пора ли завершить цикл. В результате получается более компактная и простая для восприятия инструкция.

```
<?php
```

```php
for ($num = 1; $num <= 10; $num++) {
    print "Цикл номер $num<br />\n";
}
?>
```

2.4.4 Цикл foreach

Данный цикл предназначен специально для перебора массивов. Инструкция foreach очень удобна, поскольку сама выполняет обход и чтение все элементов массива, пока не будет достигнут последний. Она позволяет не держать в памяти тот факт, что индексация массивов начинается с нуля, и никогда не выходит за пределы массива, что делает конструкцию цикла очень удобной и помогает избежать распространенной логической ошибки.

Синтаксис цикла foreach выглядит следующим образом:

```
<?php
foreach (массив as $ключ=>$значение) {
   команды;
}
?>
```

Здесь команды циклически выполняются для каждого элемента массива, при этом очередная пара ключ=>значение оказывается в переменных $ключ и $значение. Приведем пример работы цикла foreach:

```php
<?php
$shapes=array('Банка содовой' => 'Цилиндр',
    'Блокнот' => 'Прямоугольник',
    'Яблоко' => 'Шар',
    'Апельсин' => 'Шар',
    'Телефонная книга' => 'Прямоугольник');

foreach ($shapes as $key => $value) {
    # Любой ассоциативный массив хранит данные в виде пар ключ/значение
    echo $key, '- это' , $value, '<br />\n';
}
?>
```

Раз речь пошла о массивах, поговорим и о них.

2.5 Основные сведения о массивах

Чтобы работать с массивами, вам нужно освоить два новых понятия: элементы и индексы. Элементы – это значения, хранящиеся в массиве. К каждому элементу можно обратиться по его уникальному индексу. Значением индекса может быть число или строка, но это значение должно быть уникальным. Массив можно представить как

таблицу или базу данных, содержащую всего два столбца. Первый столбец уникально идентифицирует строку таблицы, а во втором хранятся значения.

Ассоциативные массивы и массивы с числовыми индексами

В массивах с числовыми индексами в качестве значений индексов используются числа, а в ассоциативных массивах – строки. В ассоциативных массивах каждому новому элементу нужно назначить уникальный строковый индекс. Массивы с числовыми индексами позволяют просто добавить элемент, а PHP автоматически назначит ему в качестве индекса первое свободное число начиная с 0. Массивы обоих типов позволяют добавлять новые элементы по одному. Ассоциативные массивы прекрасно подходят для сохранения информации о настройках, так как их ключи могут хранить смысловую информацию. Внутри PHP массивы с числовыми индексами хранятся точно так же, как и ассоциативные массивы. Массивы с числовыми индексами обеспечивают более простой способ обхода наборов данных, поскольку для доступа к следующему значению достаточно увеличить на единицу индекс предыдущего.

В качестве элементов массива могут использоваться любые значения, включая строки, числа и даже другие массивы. Поле ключа должно быть скаляром. Скалярные значения – это такие значения элементарного типа, как числа или строки, включая значения TRUE и FALSE, но не данные, которые могут иметь несколько составных значений, например массивы. Кроме того, в поле ключа массива для каждого элемента должно быть уникальное значение, иначе вы можете записать новый элемент поверх уже имеющегося, с тем же ключом. Если вы попытаетесь назначить новому элементу ключ, который уже определен для другого элемента, новое значение просто заменит старое.

Чем короче строковые значения ключей массива, тем быстрее будет работать программа, и тем проще будет сопровождать ее.

2.5.1 Присваивание через идентификатор массива

Теперь, когда вы получили основные понятия о массивах, рассмотрим способы записи значений в массив. В языке PHP записать значения в массив можно двумя способами. Сначала рассмотрим способ, основанный на идентификаторах массива.

Операция присваивания по идентификатору массива выглядит так же, как операция присваивания значения переменной, за исключением квадратных скобок ([]), которые добавляются после имени переменной массива. В квадратных скобках можно указать индекс элемента. Если индекс не указан, PHP автоматически выберет наименьший незанятый числовой индекс.

```
<?php
$weekdays[]='Понедельник';
```

```
$weekdays[]='Вторник';
?>
<?php
$weekdays[0]='Понедельник';
$weekdays[1]='Вторник';
?>
```

2.5.2 Присваивание с помощью функции array

Другой способ присваивания значений элементам массива заключается в применении специальной функции array. Функция array позволяет создавать массивы с одновременным присваиванием множества значений элементов. В качестве параметров функция принимает пары ключ/значение. Она возвращает массив, который можно присвоить переменной. Присваиваемые элементы отделяются друг от друга запятыми.

```
<?php
$weekdays=array('Понедельник',
    'Вторник',
    'Среда',
    'Пятница',
    'Суббота',
    'Воскресенье'
);
?>
```

Создание ассоциативного массива осуществляется следующим образом.

```
<?php
$shapes=array('Банка содовой' => 'Цилиндр',
    'Блокнот' => 'Прямоугольник',
    'Яблоко' => 'Шар',
    'Апельсин' => 'Шар',
    'Телефонная книга' => 'Прямоугольник');
?>
```

2.5.3 Добавление значений в массив

Добавить новые элементы в конец имеющегося массива можно с помощью его идентификатора. Например, добавить Четверг в массив $wekdays можно так:

```
<?php
    $weekdays[] = "Четверг";
?>
```

Добавить фигуру в ассоциативный массив можно с помощью аналогичного синтаксиса:

```
<?php
    $shapes["Рупор"]= "Конус";
?>
```

Такой прием работает даже в случае, если массив был создан с помощью функции array. Перед нами встает следующая задача: подсчет количества элементов массива.

2.5.4 Подсчет количества элементов в массиве

Определить текущее количество элементов в массиве можно с помощью функции count. Функция count идентична функции sizeof, они взаимозаменяемы.

```
<?php echo count($shapes);?>
```

Для функции count совершенно не важно, является ли массив ассоциативным, или это массив с числовыми индексами. Расположить элементы массива в алфавитном порядке поможет функция sort.

Функция sort() выполняет сортировку массива. По ее завершении элементы массива выстраиваются в порядке возрастания: от наименьших значений к наибольшим. Числовые значения сортируются как числа, а строки – в алфавитном порядке. Эта функция назначает элементам массива новые ключи. Она удаляет все имевшиеся ключи, которые, возможно, назначили вы, вместо того чтобы переупорядочить их.

Основной список функций для работы с массивами приведен в приложении 1.

2.6 Функции

Чтобы писать на языке PHP программы, представляющие из себя нечто большее, чем просто пара страниц кода, и при этом достаточно удобно организованные, понадобится освоить понятие функции. Функции избавят вас от многократного повторения одних и тех же фрагментов кода в программах. Чтобы получить функцию, нужно назначить фрагменту кода имя, называемое именем функции. После этого данный фрагмент можно исполнять, обращаясь к нему по этому имени.

В языке PHP очень много встроенных функций. Вы также можете создавать свои функции для организации собственного программного кода. Чтобы определить функцию, начните с инструкции function:

```
<?php
  function имя_функции([аргументы]) { программный код }
?>
```

Квадратные скобки ([]) означают необязательность. В этом определении, на месте [аргументы] можно было бы записать необязательные_аргументы. Вслед за ключевым словом function должно следовать имя функции. Имена функций подчиняются тем же правилам, что и имена других программных элементов языка PHP, например переменных. Вслед за именем функции должны следовать круглые скобки. Если у функции есть параметры, они указываются в круглых скобках. Наконец, далее должен следовать программный код функции, заключенный в фигурные скобки, как показано выше.

Функции допускается определять в любом месте программы, вызывать их тоже можно практически в любом месте.

```
<?php
```

```
    function hi() {
        echo("Привет из мира функций!");
    }
    // Вызвать функцию
    hi();
?>
```

У функции hi нет параметров, поэтому в скобках ничего не надо указывать. Теперь, когда мы рассмотрели определение простейшей функции, попробуем соединить функцию и параметры.

Параметры – это удобный способ передачи информации в функцию, позволяющий нам не заботиться об области видимости переменных. В языке PHP не требуется определять типы данных для параметров – достаточно перечислить имена параметров.

В следующем примере используется функция strtolower, переводящая буквы строки «Hello world!» в нижний регистр. Она принимает на входе строковый параметр. В примере также используется функция strtoupper, преобразующая буквы строки к верхнему регистру.

```
<?php
// Сделать заглавной первую букву строки
function capitalize( $str ) {
    // Сначала переведем все буквы в нижний регистр
    $str = strtolower($str);
    // Затем первую букву строки переведем в верхний регистр
    $str{0} = strtoupper($str{0}); // $str{0} – обращение к первому символу строки
    echo $str;
}
    capitalize("hEllo WoRld!");
?>
```

Результатом вывода на экран будет: Hello world!

Значение переменной $str выводится внутри функции, потому что мы еще не рассматривали возможность передачи значения за пределы функции. Как уже отмечалось, $str{0} – это форма обращения к первому символу строки.

2.7 Подключение PHP-файлов

Чтобы сделать программный код более удобочитаемым, вы можете поместить свои функции в отдельный файл. Многие дополнения PHP, доступные в Интернете, уже оформлены в виде файлов, которые достаточно просто можно подключить к своей программе на языке PHP. Возможность подключения файлов в PHP обеспечивают четыре функции:

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

Все эти функции могут принимать в качестве параметра имя локального файла. Функции require и include очень схожи по действию и отличаются лишь реакцией на невозможность получения запрошенного ресурса. Например, в случае недоступности ресурса функции include и include_once выводят предупреждение и пытаются продолжить исполнение программы. Функции require и require_once при недоступности запрошенного ресурса останавливают обработку данной страницы. Для примера создадим простой подключаемый файл с именем add.php.

```php
<?php
    function add( $x, $y ) {
        return $x + $y;
    }
?>
```

Подключим его к основному файлу index.php.

```php
<?php
    include("add.php");
    include("add.php"); //повторное подключение вызовет ошибку (функции add будет
инициализирована два раза
    echo add(2, 2);
?>

<?php
    include_once("add.php");
    include_once("add.php"); //ошибки не возникнет, но лучше так не делать
    echo add(2, 2);
?>
```

2.8 Cookies

Сохранять некоторые сведения о пользователе, например имя, число посещений, дату последнего посещения, можно в cookies – небольших текстовых фрагментах, размещаемых на стороне клиента; впервые cookies появились в Netscape 1.0. Информация сохраняется на машине клиента и отправляется веб-серверу с каждым новым запросом. Данные передаются вместе с HTTP-заголовками.

После первого посещения веб-сайта браузер будет возвращать серверу копию cookie при каждом соединении. В целях обеспечения безопасности прочитать cookies можно только из домена, в котором они были созданы. Кроме того, у cookies есть дата истечения срока хранения, после которой они удаляются. Максимальный объем данных, которые могут храниться в cookie, составляет 4 Кб. Отличие между cookies и сеансами состоит в том, что cookies хранятся на жестком диске клиента, тогда как информация о сеансе – на сервере. Сеансы подобны кодам, которые генерируются на основе аутентификации. Это означает, что информация о сеансе существует, только пока открыто

окно браузера (который удаляет cookie, используемый сеансом). По умолчанию сеансы задействуют единственный cookie для хранения своих кодов или идентификаторов сеанса.

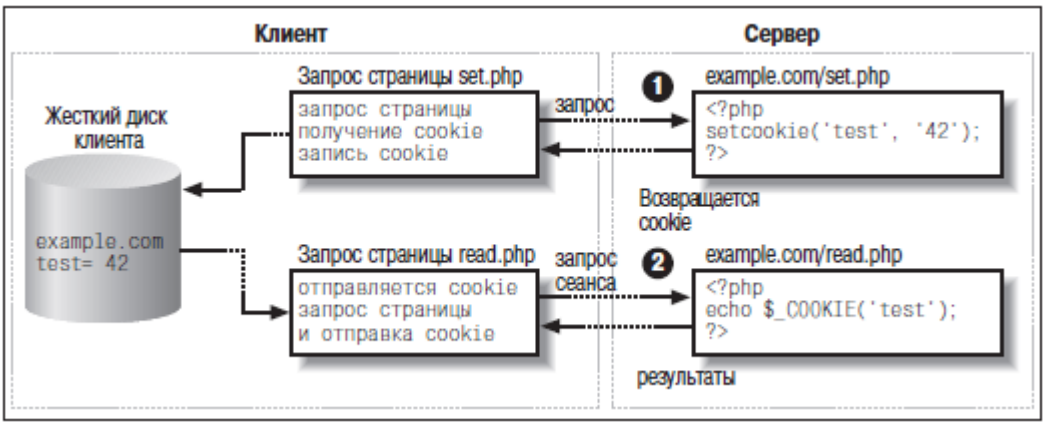

Рисунок 5 – Взаимодействие средствами Cookies

Установить cookie в PHP очень просто: для этих целей предоставляется функция setcookie. Cookies генерируются как часть заголовков HTML-страницы, поэтому очень важно вызвать функцию setcookie до того, как начнется вывод какой-либо другой информации.

Функция принимает имя cookie в качестве аргумента. Кроме того, можно указать дополнительные сведения, например:

```
setcookie (имя, значение, срок_хранения, путь, домен, режим_безопасности)
```

```php
<?php
    // Запомните: функцию setcookie следует вызвать до того,
    // как начнется вывод какой-либо другой информации
    setcookie("username","igsavenko");
?>
```

Получить значение cookie можно двумя способами. Все cookies доступны через переменную окружения $_COOKIE как $_COOKIE['имя_cookie']. Кроме того, получить значение любого cookie можно с помощью суперглобальной переменной $_SERVER[HTTP_COOKIE].

```php
<?php
if (!isset($_COOKIE['username'])) {
    echo("Cookie не установлен!");
} else {
    echo("Имя пользователя: ". $_COOKIE['username'] . ".");
}
?>
```

Уничтожить cookie можно как на стороне клиента, так и на стороне сервера. Чтобы удалить cookie на стороне клиента, достаточно отыскать в своей системе папку Cookies и удалить ее. Чтобы удалить cookies, сервер может:

– переустановить cookie, указав при этом истекший срок хранения;
– переустановить cookie, указав только его имя.

В обоих случаях используется функция setcookie. Для удаления cookie за счет указания истекшего срока хранения достаточно просто передать функции setcookie дату, которая находится в прошлом.

```php
<?php
    // Запомните: функцию setcookie следует вызвать до того,
    // как начнется вывод какой-либо другой информации
    setcookie("username","", time()-10);
    echo 'Роза';
?>
```

2.9 Сеансы (сессия)

После перехода клиента к другой странице веб-серверы по умолчанию не запоминают информацию, которая была введена на прежней странице. Это осложняет использование одной и той же информации в нескольких страницах. Сеансы (sessions) позволяют разрешить эту проблему, обеспечивая поддержку данных между страницами на протяжении всего времени посещения пользователем вашего сайта. В рамках каждого сеанса могут быть задействованы многие переменные, которые будут храниться на протяжении всего сеанса. Сервер следит за сеансами пользователей, назначая им уникальные идентификаторы, которые генерируются сервером в момент открытия сеанса. Этот идентификатор называется идентификатором сеанса (session identifier) и должен передаваться на сервер каждый раз, когда после начала сеанса запрашивается очередная страница.

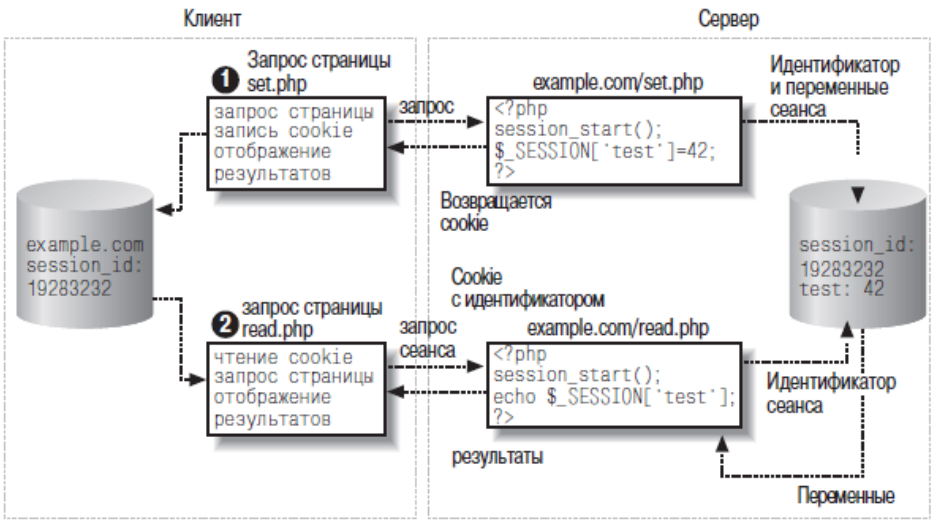

Рисунок 6 – Принцип работы сеансов

Информация о сеансах хранится на стороне сервера. Переменные сеанса сохраняются в файле, в сериализованном виде. Когда переменная сериализуется (преобразуется в последовательную форму), в файл записывается имя переменной, тип и значение в виде последовательной строки. В UNIX-подобных операционных системах этот файл обычно сохраняется в файловой системе /tmp (временная файловая система).

Чтобы начать сеанс, нужно добавить в начало PHP-сценария вызов функции session_start – лишь после этого появится возможность сохранять и получать данные, принадлежащие сеансу. Функцию session_start следует вызвать до вызова функции header и вообще до того, как браузеру клиента будет отправлена хоть какая-нибудь информация, в противном случае сеанс может работать некорректно.

```php
<?php
    session_start();
?>
```

Правильный способ создания и обращения к переменным сеанса заключается в использовании суперглобальной переменной $_SESSION с подстановкой имени требуемой переменной в квадратных скобках. Например, запись $_SESSION['variable_name'] = value; означает присваивание значения value переменной сеанса с именем variable_name.

```php
<?php
    session_start();
    $_SESSION['hello'] = 'Hello World';
    echo $_SESSION['hello'];
?>
```

Есть моменты, когда требуется преждевременно завершить сеанс. Например, если вы разместили на странице кнопку или гиперссылку выхода из системы. Выход из системы фактически означает завершение сеанса работы с пользователем. Завершают сеанс с помощью функции session_destroy. Разумеется, сеанс предварительно должен быть открыт, чтобы было что завершать.

2.10 Взаимодействие с MySQL

2.10.1 Подключение к БД

Прежде всего, необходимо выполнить подключение к базе данных и убедиться в том, что соединение установлено. Подключив файл с информацией о соединении с базой данных, мы получаем возможность использовать переменные в вызове функции mysql_connect вместо жестко определяемых параметров. Здесь мы просто объединили файл db_test.php и небольшой фрагмент программного кода.

```php
<?php
    // Подключить файл с информацией о соединении с базой данных
    include('db_login.php');
    // Подключиться к базе данных
    $connection = mysql_connect($db_host, $db_username, $db_password);
    if (!$connection) {
        die("Невозможно подключиться к базе данных: <br />". mysql_error());
    }
?>
```

Функция mysql_connect принимает в качестве аргументов имя хоста, имя пользователя и пароль. Если соединение было благополучно установлено, функция возвращает дескриптор соединения. Если соединение не может быть установлено, возвращается значение FALSE. Чтобы убедиться в том, что соединение установлено, нужно проверить возвращаемое значение. Если при подключении была обнаружена какая-либо ошибка, например неверный пароль, следует вывести предупреждение с помощью функции mysql_error, указав причину ошибки.

Следующий этап после установления соединения – выбор используемой базы данных с помощью функции mysql_select_db. Она принимает два аргумента: имя базы данных и необязательный дескриптор соединения. Если дескриптор не указан, по умолчанию будет использовано соединение, установленное последним вызовом функции mysql_connect.

```php
<?php
    // Выбрать базу данных
    $db_select = mysql_select_db($db_database);
    if (!$db_select) {
        die("Невозможно выбрать базу данных: <br />". mysql_error());
    }
?>
```

2.10.2 Исполнение запроса

Построить SQL-запрос не сложнее, чем записать в некоторую переменную строку с текстом запроса. Разумеется, SQL-запрос должен быть построен правильно, в противном случае MySQL вернет сообщение об ошибке при попытке исполнить его.

Исполнение запросов к базе данных производится с помощью функции mysql_query. Она принимает два аргумента – запрос и необязательный дескриптор соединения с базой данных, и возвращает результат запроса. Сохраним ссылку на результат в переменной с говорящим именем $result. Эту переменную также следует проверять на равенство значению FALSE, чтобы убедиться в отсутствии ошибок в строке запроса или в соединении с базой данных.

```php
<?php
    // Записать текст запроса в переменную
    $query = "SELECT * FROM books NATURAL JOIN authors";
    // Исполнить запрос
    $result = mysql_query( $query );
    if (!$result) {
        die("Невозможно исполнить запрос к базе данных: <br />". mysql_error());
    }
?>
```

Исполнив запрос, база данных возвращает результирующий набор данных. Это такие же строки, как при исполнении запросов с помощью клиента командной строки mysql. Чтобы вывести их, придется обрабатывать каждую строку отдельно.

Извлечь строки из результирующего набора данных можно с помощью функции mysql_fetch_row. Ее синтаксис: array mysql_fetch_row (resource $result);

Данная функция принимает в качестве аргумента результат исполнения запроса, который мы сохранили в переменной $result. При каждом вызове она возвращает одну строку, пока не достигнет последней записи в результирующем наборе данных, после чего будет возвращать значение FALSE. Таким образом, нужно организовать выборку данных в цикле с помощью функции mysql_fetch_row и определить некоторый программный код, выводящий каждую строку:

```php
<?php
    // Получить и отобразить результаты
    while ($result_row = mysql_fetch_row(($result))){
        echo 'Название: '.$result_row[1] . '<br />';
        echo 'Автор: '.$result_row[4] . '<br /> ';
        echo 'Страниц: '.$result_row[2] . '<br /><br />';
    }
?>
```

Записи в результирующем наборе данных хранятся в виде массивов, и за одну операцию можно извлечь только одну строку. Конструкция $result_row[2] означает обращение ко второму полю (в соответствии с порядком следования столбцов в запросе или в определении таблицы при использовании запроса вида SELECT *) строки из результирующего набора данных.

Как правило, по окончании работы с базой данных следует закрыть соединение. Это можно осуществить с помощью функции mysql_close, которая сообщит PHP и MySQL, что вы больше не будете использовать соединение, и освободит все занимаемые им ресурсы и память.

Обязательно изучить синтаксис оператора SELECT.

http://www.php.su/mysql/manual/?page=SELECT

2.10.3 Добавление данных

Чтобы добавить информацию о покупке во вновь созданную таблицу, выполним запрос с инструкцией INSERT.

```
<?php
    $query = "INSERT INTO authors VALUES (NULL,$last_value,'Alex Martelli')";
    $result = mysql_query($query);
    if (!$result) {
        die("Ошибка исполнения запроса к базе данных: <br />".$query." "
.mysql_error());
    }
    echo "Обновление успешно выполнено!";
?>
```

Обязательно изучить синтаксис оператора INSERT,

http://www.php.su/mysql/manual/?page=INSERT

2.10.4 Обновление данных

Вы можете изменять записи в таблицах с введенными данными. Скорее всего, это потребуется только для исправления ошибок в данных или внесения изменившихся сведений о пользователе.

```
<?php
    $query = "UPDATE books SET pages=558 WHERE title_id=2";
    $result = mysql_query($query);
    if (!$result) {
        die("Ошибка исполнения запроса к базе данных: <br />".$query." " .
mysql_error());
    }
    echo "Обновление успешно выполнено!";
?>
```

Если требуется одновременно обновить содержимое нескольких полей в одной и той же записи, вы должны разделить код запятыми (,). Здесь также можно использовать динамические значения, например подставить в предложение WHERE данные из формы. Предложение WHERE указывает, какие записи необходимо обновить; при отсутствии этого предложения изменения будут произведены во всех записях.

Обязательно изучить синтаксис оператора UPDATE,

http://www.php.su/mysql/manual/?page=UPDATE

2.10.5 Удаление данных

Удалить имеющиеся в базе данных записи позволяет команда DELETE. Помните: после выполнения операции удаления данные невозможно восстановить – они утеряны

навсегда. Убедитесь, что при удалении имеющихся данных выполняются все необходимые проверки. Обязательно используйте предложение WHERE, чтобы не удалить из таблицы все записи.

```php
<?php
    // Записать текст запроса в переменную
    $query = "DELETE FROM books WHERE authors = 1";
    // Исполнить запрос
    $result = mysql_query( $query );
    if (!$result) {
        die("Невозможно исполнить запрос к базе данных: <br />". mysql_error());
    }
?>
```

Обязательно изучить синтаксис оператора DELETE, http://www.php.su/mysql/manual/?page=DELETE.

В данном методическом пособии приведены только базовые конструкции и возможности языка PHP. В методическом пособии не рассматривается Объектно-ориентированная составляющая языка PHP, аутентификация, безопасность, регулярные выражения и многое другое. Полученных знаний Вам будет достаточно для выполнения лабораторных работ и разработки простых веб-страниц.

2.11 **Лабораторная работа 10.** Основы PHP

Перед тем, как перейти к разработке «Гостевой книги» применим полученные в выше описанной теоретической главе знания на практике.

Задание 1.

1. Запустите OpenServer.
2. В папке …OpenServer\domains\localhost создайте папку с Вашей фамилией на английском языке, например, «ivanov». Затем в созданной директории создайте папку «lab10»
3. Используя, удобную для Вас среду разработки: Eclipse, Netbeans PHP, PHPStorm или другую, создайте в директории файл index.php и файл data.php.

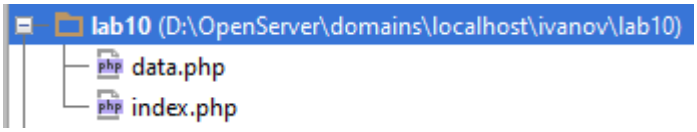

Рисунок 7 – Результат создания файлов

4. Следующим шагом создайте в файле data.php массив и подключите (изучив внимательно функцию **`include_once`** – возможно Вам придется воспользоваться дополнительной литературой) его к файлу index.php. Пример массива приведен ниже. Выведите массив используя функцию `print`. Посмотрите результат в браузере http://localhost:88/ivanov/lab10/index.php. В случае возникновения ошибки исправьте ее и не забудьте сделать выводы в отчете.

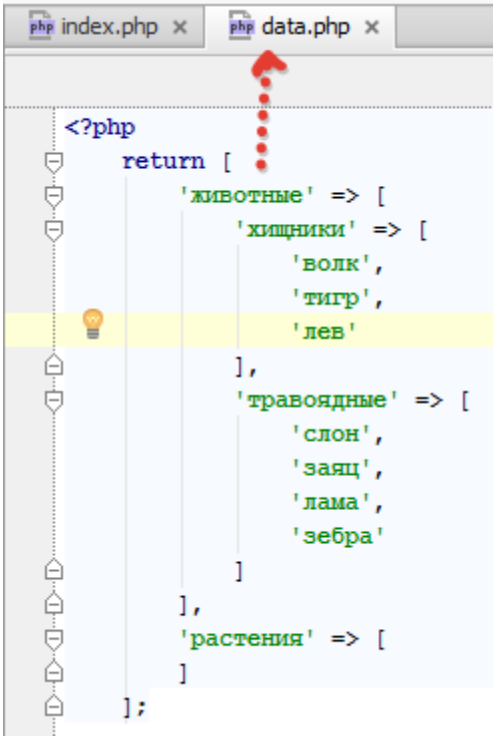

Рисунок 8 – Пример массива

5. Используя цикл `foreach` выведите данный массив в виде вложенного списка, результат HTML-страницы представлен ниже.

```
<ul>
    <li>животные
        <ul>
            <li>хищники
                <ul>
                    <li>волк</li>
                    <li>тигр</li>
                    <li>лев</li>
                </ul>
            </li>
            <li>травоядные
                <ul>
                    <li>слон</li>
                    <li>заяц</li>
                    <li>лама</li>
                    <li>зебра</li>
                </ul>
            </li>
        </ul>
    </li>
    <li>растения
        <ul></ul>
    </li>
</ul>
```

Задание 2.

1. Усложним задачу. Создайте в файле index.php функцию с названием menuFormatter(), которая осуществляет создание меню созданного Вами в задании 1.
2. Вызовите созданную функцию следующим образом:
```
<!DOCTYPE html>
<html>
    <head>
        <title>Лабораторная работа</title>
    </head>
    <body>
        <?php menuFormatter(); ?>
    </body>
</html>
```
3. Подключите к файлу index.php библиотеки Bootstrap и jQuery. Используя возможности bootstrap добавьте возможность разворачивать элементы первого уровня меню. Для этого Вам понадобятся следующие возможности jQuery. Меню должно располагаться вертикально с использованием панелей.
   - http://getbootstrap.com/components/#panels
   - http://getbootstrap.com/components/#nav-pills
   - http://getbootstrap.com/javascript/#collapse

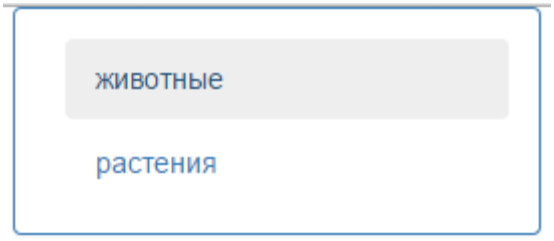

Рисунок 9 – Меню до разворачивания

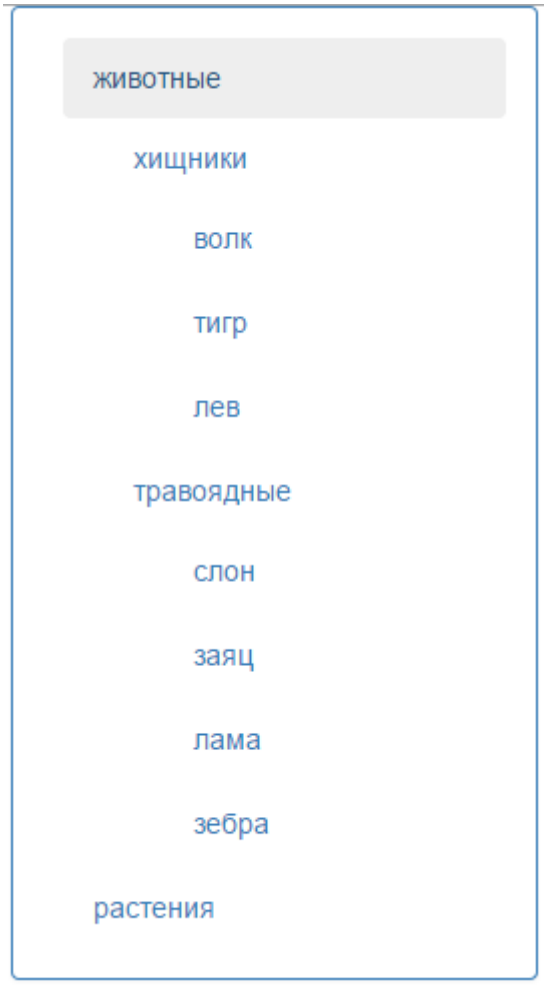

Рисунок 10 – Меню после разворачивания

Задание 3. Взаимодействие с БД

1. Создайте базу данных содержащую три таблицы
    a. regnum (содержит царство: животное, растение, грибы и т.д.)
    b. ordines (отряд: хищники, травоядные, китообразные и т.д.)
    c. species (вид: заяц, волк и т.д.)
2. Не забудьте продумать связь между таблицами, чтобы определить всю иерархию для построения меню (аналогичного второму заданию).
3. Добавьте данные в таблицы используя phpMyAdmin.
4. Осуществите подключение к БД средствами PHP.
5. Извлеките данные из БД, сформировав аналогичный массив данных как в задании 1.

2.12 Лабораторная работа 11. Пример разработки гостевой книги

2.12.1 Функционал гостевой книги

Перед началом разработки определимся, какие функции будет выполнять гостевая книга:

- вход для администратора, имя пользователя и пароль будет прописан в коде;
- обеспечивает хранение сессии для администратора, что при следующем входе на сайт ему не пришлось повторно авторизоваться;
- администратор может управлять записями пользователя (удалять, модерировать);
- пользователь может добавлять сообщения, но они будут добавляться только после модерации;
- пользователь может загружать на сервер не более одного изображения;
- предотвращать автоматическое добавление сообщений специализированными скриптами используя CAPTCHA.

2.12.2 Эскиз гостевой книги

После того, как мы определили функционал, можно спроектировать эскиз будущей гостевой книги.

Рисунок 11 – Эскиз гостевой книги

2.12.3 Создание таблицы в MySQL

Выше представленный эскиз охватывает все требования, предъявляемые к гостевой книге, соответственно можно приступить к разработке. Для начала определим данные, которые будут храниться в БД. Для хранения новостей достаточно одной таблицы «Новости» со следующими атрибутами:

- имя пользователя;
- e-mail пользователя;
- текст сообщения;
- дата добавления сообщения;
- опубликовано/не опубликовано;
- путь до загруженного на сервер изображения;

В третьей лабораторной работе, вы уже проектировали БД в СУБД MySQL и средой проектирования БД MySQL Workbench, поэтому я приведу только результат (рисунок 12).

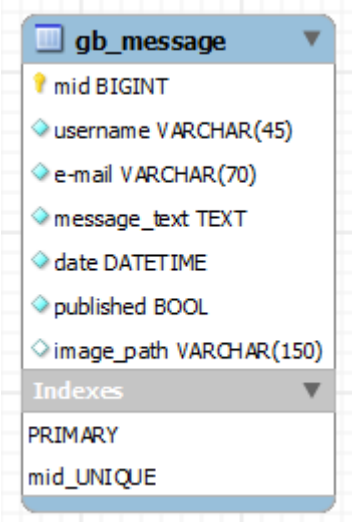

Рисунок 12 – Таблица сообщений гостевой книги

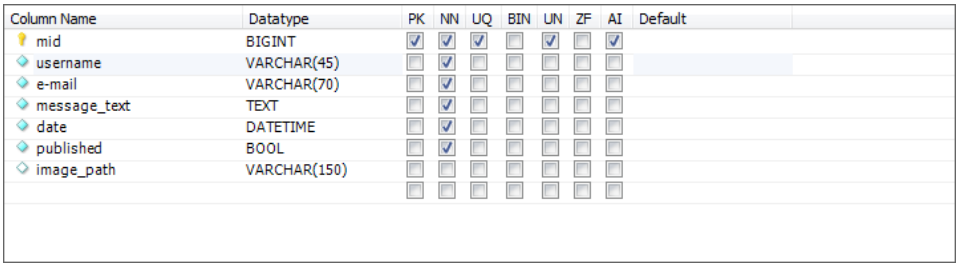

Рисунок 13 – Описание атрибутов

Ниже приведен автоматически сгенерированный код на языке SQL для создания таблицы «gb_message» в БД с именем «gusetbook».

```
CREATE TABLE IF NOT EXISTS `guestbook`.`gb_message` (
  `mid` BIGINT(19) UNSIGNED NOT NULL AUTO_INCREMENT,
  `e-mail` VARCHAR(70) CHARACTER SET 'utf8' COLLATE 'utf8_general_ci' NOT NULL,
  `message_text` TEXT CHARACTER SET 'utf8' COLLATE 'utf8_general_ci' NOT NULL,
  `date` DATETIME NOT NULL,
  `published` TINYINT(1) NOT NULL,
  `image_path` VARCHAR(150) NULL DEFAULT NULL,
  PRIMARY KEY (`mid`),
  UNIQUE INDEX `mid_UNIQUE` (`mid` ASC))
ENGINE = InnoDB
```

```
DEFAULT CHARACTER SET = utf8
COLLATE = utf8_general_ci;
```

На этом создание таблицы закончено, можно перейти к верстке будущей гостевой книги.

2.12.4 Верстка веб-страницы

Создадим документ index.html и убедимся, что кодировка документа совпадает с настройками веб-сервера, в инструкции по установки локального веб-сервера мы выбирали кодировку utf-8, соответственно кодировка файла и кодировка для браузера (указанная в мета-тегах) должна быть utf-8.

```
<meta http-equiv="content-type" content="text/html; charset=utf-8">
```

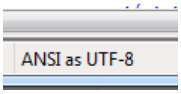

Рисунок 14 – Кодировка файла

Результат верстки приведен на рисунке 15, исходные коды находятся в приложениях

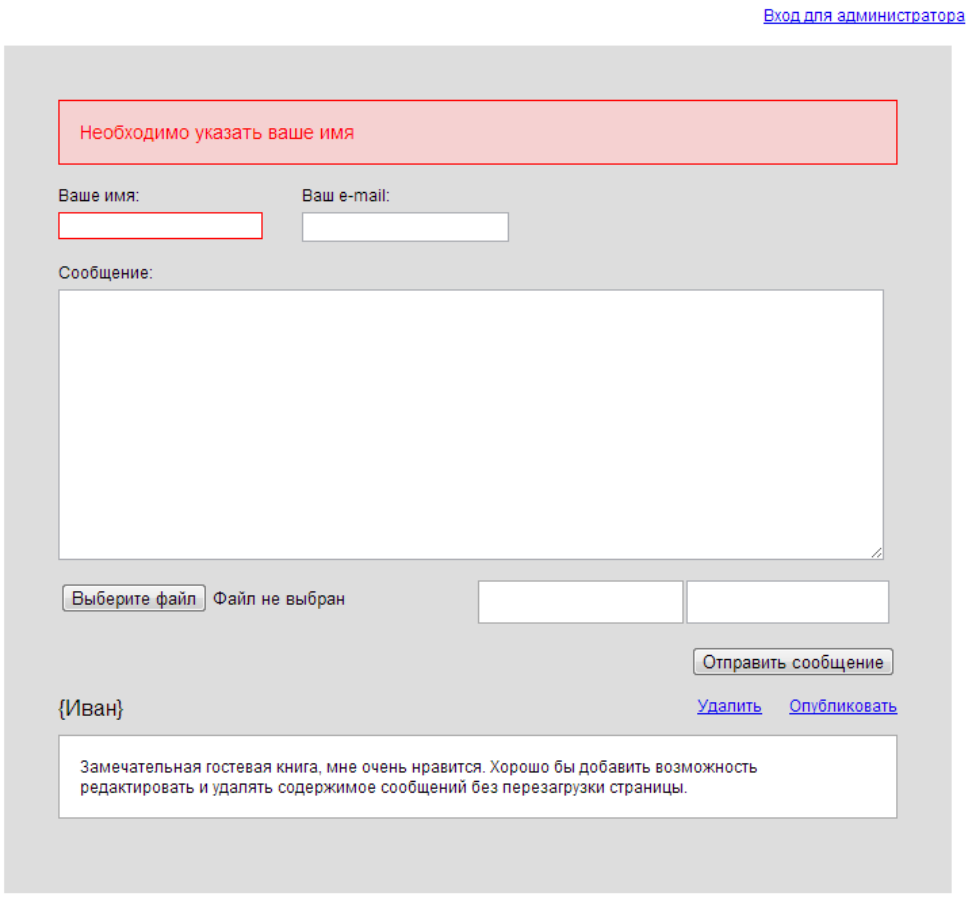

Рисунок 15 – Результат верстки

Создадим в корневой директории виртуального хоста (для OpenServer директория «project», которая была создана в инструкции) на веб-сервере следующие директории:

- css – данная директория будет содержать файлы каскадных таблиц стилей;
- files – данная директория будет содержать загруженные изображения пользователей;
- images – изображения необходимые для верстки.

Так же скопируем туда результат нашей верстки файл index.html и файл со стилем. Перейдем к самому интересному непосредственно реализации гостевой книги, первым делом реализуем основную функцию гостевой книги это добавление новости в БД, затем добавим извлечение записей из БД, для проверки, что записи добавляются корректно. Затем расширим функционал гостевой книги, добавив загрузку файлов и проверку от автоматического добавления записей (CAPTCHA) и завершающим штрихом будет реализация авторизации администратора на сайте с возможностью удаления и публикации сообщений.

2.12.5 Реализация гостевой книги

Перед тем, как перейти к разработке с использованием языка PHP, необходимо изменить расширение файл index.html на *.php. Таким образом, веб-сервер будет передавать данный файл на обработку интерпретатору PHP.

2.12.5.1 Добавление новости в БД

Первым делом давайте рассмотрим, как осуществляется передача данных, введённых пользователем в форме добавления сообщения, на сервер. При создании формы в HTML-коде мы указали, что передача будет осуществляться при помощи метода POST, так как размер передаваемых данных будет достаточно велик, а обработку значений будет проводить этот же скрипт «index.php».

```
<form method="POST" action="index.php">
  . . .
  <input type="text" id="gb_user_name" class="error" name="gb[username]">
  . . .
</form>
```

Переданные от клиента данные в скрипт «index.php» будут храниться в глобальном массиве $_POST, подробнее про массивы вы можете прочитать в разделе «Основы языка PHP». В качестве ключа в массиве будет являться имя (атрибут name) текстового поля, которое мы указали при верстке, но обратите внимание, что в качестве значения у атрибута name тега input указан массив: gb[username]. Это значит, что внутри массива $_POST будет еще один массив с именем gb. Это сделано для того, чтобы логически разделить основные данные (Имя, E-mail, сообщение) от дополнительных (CAPTCHA, файл).

Добавим конструкцию <?php print_r($_POST);?> в файл «index.php», которая отобразит содержимое массива. В результате на экран будет выведено следующее:

```
Array
(
   [gb] = Array
      (
         [username] = any
         [usermail] = any@any.com
         [message] = Any text
      )

   [captcha] = “”;
)
```

Соответственно, чтобы получить имя пользователя, который оставил комментарий необходимо написать следующее: <?php echo $_POST[“gb”][“username”];?>, что касается CAPTCHA, то <?php echo $_POST[“captcha”];?>.

Если вы передаете данные на сервер используя метод GET, то значения будут содержаться в глобальном массиве $_GET.

Не достаточно просто передать данные на сервер и сразу записывать их в БД, перед записью необходимо их обработать и проверить на корректность. Как пример можно проверить, что пользователь ввел корректный e-mail адрес, а из текста удалить HTML теги, иначе пользователь может испортить нам верстку, добавив собственные теги.

Проверка электронной почты на корректность

Для проверки корректности электронного почтового ящика можно использовать различные способы. Самыми распространенными являются регулярные выражения или фильтры. Регулярные выражения являются сложным, но очень полезным и более

надежным механизмом проверки или разбора содержимого, в данном курсе мы не будем уделять им внимания.

Рассмотрим стандартный механизм проверки данных, который появился в версии PHP 5.0 – фильтры.

Фильтры разделены на 3 типа:

- Validate Filters - проверяют соответствие строки фильтру. Ответ от такого фильтра - исходная строка в случае удачи или false;
- Sanitize Filters - очищающие фильтры. Обрабатывают строку и возвращают её в отфильтрованном виде;
- FILTER_CALLBACK – фильтр передаст строку пользовательской функции и вернет её ответ.

Основная функция для работы с фильтрами:

mixed filter_var (mixed $variable [, int $filter = FILTER_DEFAULT [, mixed $options]])

Первый параметр - переменная, подлежащая проверки, второй - номер фильтра, который удобно записывается предопределенной константой. Третьим параметром можно передать дополнительные опции, как правило - специальные флаги фильтров. Возвращаемое значение зависит от фильтра.

Рассмотрим конкретные примеры:

```php
<?php
    if(filter_var($_POST["gb"]["usermail"], FILTER_VALIDATE_EMAIL)){
        echo $_POST["gb"]["usermail"] . ' - Ваш e-mail корректный.';
    }else{
        echo $_POST["gb"]["usermail"] . ' - У вас не корректный e-mail';
    }
    echo filter_var($_POST["gb"]["usermail"], FILTER_SANITIZE_EMAIL);
?>
```

На экран будет выведено следующее

(any)@any.com - У вас не корректный e-mail. //e-mail не корректный
any@any.com //удалены не корректные символы из адреса электронной почты

Любые данные переданные от клиента на сервер так же стоит проверить на наличие спецсимволов, соответствие ожидаемым типам данным, а также на наличие HTML-тегов.

Реализуем функцию, которая будет осуществлять проверку входных данных и возвращать массив ошибок.

```php
<?php
    $globMessages = array(); //массив с сообщениями
    //переменные в которых будут содержаться переданные данные (изначально пустые)
    $userName = "";
    $userMail = "";
    $userMessage = "";

    function checkData($userName, $userMail, $userMessage) {
        $errors = array();
        if(empty($userName)) {
                $errors["username"] = "Анонимные сообщения не принимаются";
        }
        if(empty($userMail) || !filter_var($userMail, FILTER_VALIDATE_EMAIL)) {
            $errors["usermail"] = "Не корректный электронный почтовый адрес";
        }
        if(empty($userMessage)) {
            $errors["message"] = "Пустое сообщение";
        }
        return $errors;
    }
    //если какие-то данные пришли от клиента и их количество совпадает
    if(isset($_POST["gb"]) && count($_POST["gb"]) == 3) {
    //фильтруем данные
    $userName = addslashes($_POST["gb"]["username"]);
    $userMail = addslashes($_POST["gb"]["usermail"]);
    $userMessage = strip_tags(addslashes($_POST["gb"]["message"]));
    //вызываем функцию проверки, которая возвращает массив сообщений об ошибках
    $globMessages = checkData($userName, $userMail, $userMessage);
    }
?>
```

Далее выводим список сообщений, в специальном блоке, который был определен при составлении эскиза.

```php
<div class='system_message
    //если есть сообщения, значит добавляем класс warning_message (возникла ошибка)
    <?php echo (!empty($globMessages)) ? "warning_message" : "success_message" ;?> '>
    <?php
        if(!empty($globMessages)) {
            echo "<ul>";
            //в цикле выводим сообщения, записанные в массиве
            foreach($globMessages as $field => $message) {
                echo "<li>".$message."</li>";
            }
            echo "</ul>";
        }
    ?>
</div>
```

После отправки формы на сервер, содержимое формы стирается, что бы это исключить, нам нужно записывать полученные данные обратно в поля ввода, для этого используется атрибут value.

```php
<input type="text" id="gb_user_name" value="<?php echo $userName;?>">
```

Так же в эскизе указано, что необходимо подсвечивать красным некорректные поля ввода, для этого мы завели специальный стиль .error для текстовых полей ввода.

```
<input type="text" id="gb_user_name"
//проверяем есть ли в сообщения ошибка для этого текстового поля, если есть добавляем класс error
<?php echo array_key_exists("username", $globMessages)? "class='error'" : "";?>
name="gb[username]" value="<?php echo $userName;?>">
```

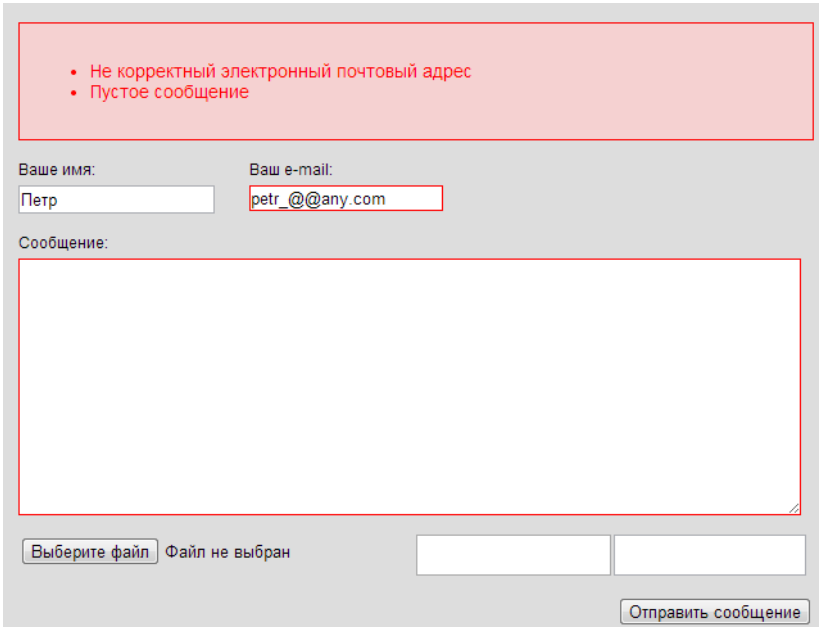

Рисунок 16 – Результат проверки введеных данных

Удаление HTML-тегов из текста сообщения

Удаление осуществляется при помощи специальной функции strip_tags, данная функция позволяет не только отчистить текст от тегов, но также может и пропускать разрешенные разработчиком теги.

```
<?php
    echo strip_tags($_POST["gb"]["message"]); //удаляет все теги
    echo strip_tags($_POST["gb"]["message"], '<a>'); //удаляет все теги, за исключением <a>
?>
```

После того, как мы научились фильтровать и обрабатывать данные можно перейти к записи данных в БД, перед этим нужно осуществить подключение к БД.

Подключение к базе данных

Интерпретатор PHP сам выполняет подключение к базе данных и позволяет сразу же начать исполнение запросов и сбор данных. Для подключения к базе данных нам потребуется:

- IP-адрес сервера базы данных;
- имя базы данных;
- имя пользователя базы данных;
- пароль.

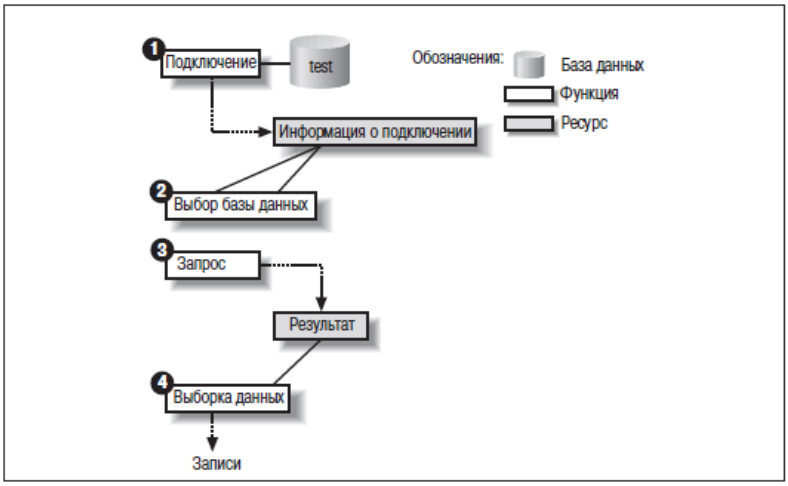

Рисунок 17 – Взаимодействие с БД для выборки данных

Добавим в файл с настройками имя пользователя, пароль, имя базы данных и адрес сервера, где расположена БД. Обращение к БД будет практически во всех скриптах, чтобы исключить дублирование кода стоит вынести подключение в отдельный файл, назовем его «bd_connect.php». Функция mysql_connect возвращает дескриптор соединения с MySQL в случае успешного выполнения или false в случае возникновения ошибки. Функция mysql_select_db() выбирает для работы указанную базу данных на сервере, на который ссылается переданный указатель.

По окончанию работы с БД, соединение стоит закрыть.

```
<?php
	mysql_close($id_connect);
?>
```

Запись сообщения в БД

Создадим переменную, которая будет сигнализировать о процессе добавления новой записи, переменная может принимать три значения:

- 0 - данные еще не добавлялись;

- 1 - данные успешно добавлены;
- -1 - при добавлении возникли ошибки.

Функция addData добавляет данные в БД.

```
<?php
function addData($userName, $userMail, $userMessage) {
    //формируем SQL-запрос
    $query = "INSERT INTO `gb_message` (`username`,`e-
mail`,`message_text`,`date`,`published`)".
        "VALUES('".$userName."','".$userMail."','".$userMessage."',NOW(),0)";
    //выполняем запрос на добавление в БД
    $queryResult = mysql_query($query, $id_connect);
    //Если возникла ошибка, то добавляем сообщение в массив, а переменную
устанавливаем в -1
    if(!$queryResult) {
        $globMessages['insert_error'] = 'Ошибка добавления: ' . mysql_error();
        return -1;
    //иначе добавляем сообщение об успешном добавлении и переменную устанавливаем в 1
    } else {
        $globMessages['success'] = 'Данные успешно добавлены, идентификатор вашей
записи: '. mysql_insert_id(); //выводит последний добавленный номер сообщения
        return = 1;
    }
}

if(empty($globMessages)) {
    //вызов функции
    $result = addData($userName, $userMail, $userMessage);
} else {
    $result = -1;
}

?>
```

В результате добавления данного участка кода осуществляется добавление записи в БД, но только в случае успешной проверки входных данных. Результат добавления выглядит вот так:

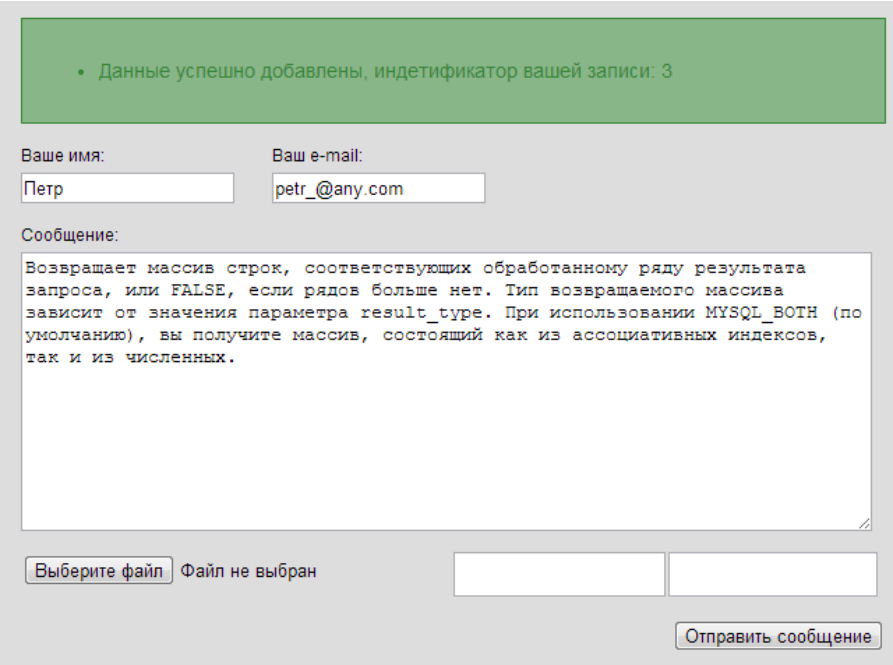

Рисунок 18 – Успешное добавление сообщения

Чтобы проверить результат необходимо извлечь данные из БД.

Вывод добавленных сообщений

Следующим шагом будет извлечение всех добавленных в БД записей пользователей, для этого мы воспользуемся запросом для извлечения данных SELECT.

Для вывода содержимого из БД, нам достаточно сделать соответствующий запрос и в цикле выводить ранее свёрстанное представление сообщения, подставив извлеченные данные.

Запрос «SELECT * FROM `gb_message`», извлекает все записи из таблицы, в дальнейшем этот запрос будет доработан, чтобы извлекать только опубликованные записи в гостевой книги. Функция mysql_fetch_array возвращает одну запись из таблицы в виде ассоциативного массива. В цикле просматривается весь список записей, на каждой итерации из ассоциативного массива извлекается одна запись в переменную $row. Обращение к значению осуществляется по имени атрибута в таблице.

```
<?php
    $queryResult = mysql_query("SELECT * FROM `gb_message`");
    while ($row = mysql_fetch_array($queryResult, MYSQL_BOTH)) {
?>
    <div class="gb_message">
        <div class="gb_mess_info">
            <div class="gb_mess_author"><?php echo $row['username'];?></div>
            <!--Данный блок будет виден только администратору-->
            <ul class="gb_mess_control">
```

```
                <li><a href="mes_control.php?action=del&mid=<?php echo
$row['mid'];?>">Удалить</a></li>
                <li><a href="mes_control.php?action=publish&mid=<?php echo
$row['mid'];?>">Опубликовать</a></li>
            </ul>
        </div>
        <div class="gb_mess_content">
            <div class="future_mess_image"></div>
            <?php echo $row["message_text"];?>
        </div>
    </div>
<?php
      }
?>
```

Представьте имя пользователя в виде ссылки (на его e-mail) по нажатию на которую, должен открываться почтовый клиент для написания электронного письма пользователю. Также рядом с именем добавьте дату публикации сообщения. Добавьте визуальное отличие для опубликованных сообщений, проверенное администратором сообщение должно быть в зеленой рамке.

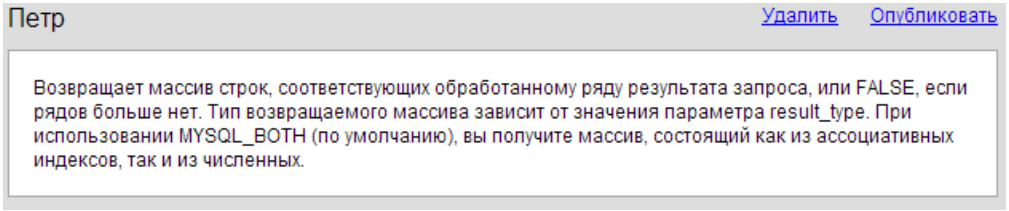

Рисунок 19 – Добавленное сообщение

2.12.5.2 Авторизация администратора

Как мы уже ранее решили, мы не будем заводить отдельную таблицу для пользователей, логин и пароль администратора будет заранее определен в исходном коде. Первым делом необходимо создать файл «admin.php», рядом с файлом «index.php». Файл admin.php будет содержать в себе форму авторизации, а также функции проверки подлинности введенных пользователем данных. Логин и пароль мы вынесем отдельно, в файл с настройками гостевой книги «config.php».

Рассмотрим форму (в документе admin.php), которая будет передавать данные о пользователе на сервер, как вы можете заметить, данные из формы будут переданы в этот же документ – «admin.php», в этом же документе они и будут обрабатываться. Данные передаются методом POST. У полей ввода есть атрибут name, который задает имя поля. При передачи данных на сервер, данные содержащиеся в этих полях ввода будут содержаться в глобальном массиве $_POST.

```
<form action="admin.php" method="POST">
    <div>
        <label for="admin_name">Логин</label>
        <input id="admin_name" type="text" name="admin_name" />
    </div>
    <div>
        <label for="admin_pass">Пароль</label>
        <input id="admin_pass" type="password" name="admin_pass" />
    </div>
    <input type="submit" value="Login"/>
</form>
```

Таким образом, в сценарии «admin.php» мы можем получить доступ к значениям следующим образом: $_POST[‘admin_name’] и $_POST[‘admin_pass’]. Используя конструкцию <?php print_r($_POST); ?> будет выведено содержимое массива $_POST, пример:

```
Array (
     [admin_name] => admin
     [admin_pass] => admin
 )
```

Как уже было сказано выше, мы создадим отдельный файл с настройками, куда вынесем учетные данные администратора, ниже приведено содержимое файла «config.php».

```
<?php
    define("USERNAME", "gb_admin");
    define("PASSWORD", "gb_password");
?>
```

В данном файле создаются константы и задаются значения, имя констант следует задавать прописными буквами, так как это будет отличать их от обычных переменных, как вы знаете значения констант изменить нельзя в отличии от переменных. Чтобы подключить данный файл к файлу «admin.php» необходимо использовать следующую конструкцию:

```
<?php
    include_once("config.php");
?>
```

Мы можем с легкостью проверить, подключился ли данный файл или нет, для этого достаточно вывести содержимое константы на экран:

```
<?php
    if(defined("USERNAME")) {
        echo USERNAME;
    }
?>
```

Теперь осталось проверить совпадает ли учетные данные, описанные в файле «config.php» с данными, которые были введены пользователем, завести сессию в случае

успешного ввода и перенаправить администратора на главную страницу. Подробно о сессиях и функциях и empty и isset вы можете прочитать в разделе «Основы языка PHP»

```
<?php
    session_start(); //начинаем работу с сессией
    /*При повторном обращении к admin.php проверяем заведена сессия или нет*/
    if(isset($_SESSION["admin"])) {
        /*Если заведена сообщаем об этом пользователю*/
        echo "Вы уже авторизированы как " . $_SESSION["admin"] . ". Вы будете
перенаправлены на главную страницу через 2 секунды.";
        /*Делаем перенаправление через 2 секунды на главную страницу*/
        header('Refresh: 2; URL=/index.php');
        exit;
    }
?>

<?php
    /*Убедимся, что данные переменные существуют*/
    if(isset($_POST['admin_name']) && isset($_POST['admin_pass'])) {
        /*и что они не пустые*/
        if(empty($_POST['admin_name']) || empty($_POST['admin_pass'])) {
            echo "Извините, вы должны заполнить все поля";
        } else {
            /*Проверяем совпадение логина и пароля со значениями констант*/
            if($_POST['admin_name'] == USERNAME && $_POST['admin_pass'] == PASSWORD)
{
                /*Если совпало заносим имя пользователя в сессию*/
                $_SESSION["admin"] = "gb_admin";
                /*Делаем переадресацию на главную страницу*/
                header('Location: /index.php');
                exit;
            } else {
                echo "Извините, пара логин/пароль не корректны";
            }
        }
    }
?>
```

После того, как пользователь введет корректные данные будет заведена сессия, а в глобальном массиве будет содержать его логин. Добавьте на главную страницу ссылку, которая позволит удалить сессию администратора.

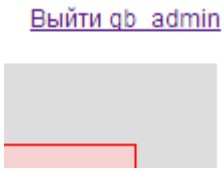

Рисунок 20 – Ссылка для удаления сессии

2.12.5.3 Управление сообщениями (данная возможность доступна только администратору)

Для управления записями в гостевой книги создадим еще один php-файл «mes_control.php». В данный файл методом GET будет передаваться действие, которое нужно над записью провести: удалить или опубликовать, а также уникальный идентификатор записи, полученный из базы. Так же не стоит забывать, что данные действия доступны только администратору, поэтому необходимо добавить соответствующую проверку.

```
<?php
    session_start();
    include_once("db_connect.php");
    //убеждаемся что администратор авторизовался
    if(isset($_SESSION["admin"])) {
        //проверяем входные данные (номер сообщения и действие)
        if(isset($_GET["action"]) && isset($_GET["mid"]) && is_numeric($_GET["mid"]))
{
            $mid = $_GET["mid"];
            $action = $_GET["action"];
            switch($action) {
                case "del":
                    //удаляем сообщение по уникальному идентификатору
                    $queryDelete = "DELETE FROM `gb_message` WHERE mid = ". $mid;
                    $queryResult = mysql_query($queryDelete);
                    if($queryResult) {
                        //возвращаемся на главную страницу и передаем методом GET о
том, что сообщение было удалено успешно
                        header('Location: /index.php?mesDelete=true');
                        exit;
                    }
                break;
                case "publish":
                    $queryUpdate = "UPDATE `gb_message` SET published = 1 WHERE mid =
".$mid;
                    $queryResult = mysql_query($queryUpdate);
                    if($queryResult) {
                        header('Location: /index.php?mesUpdate=true');
                        exit;
                    }
                break;
                default:
                    echo "Неизвестное действие.";
                    header('Refresh: 2; URL=/index.php');
                }
            }
    } else {
        echo "Управлять сообщениями может только администратор, вы будете
перенаправлены на главную страницу через 2 секунды.";
        header('Refresh: 2; URL=/index.php');
        exit;
    }
?>
```

Как вы могли заметить, при помощи функции header мы перенаправляем пользователя обратно на главную страницу, но так же передаем и сигнал (в виде переменной с установленным значением true) о том, что удаление или обновление

сообщения прошло успешно. Для обработки и отображения сообщения нужно немного модифицировать главную страницу гостевой книги, добавив:

```php
<?php
    //Выведем сообщение если удаление или обновление сообщения прошло успешно
    if(isset($_GET["mesDelete"]) && $_GET["mesDelete"] == true) {
        $globMessages['mesDelete'] = "Сообщение было удалено";
        $result = 1;
    }else if(isset($_GET["mesUpdate"]) && $_GET["mesUpdate"] == true) {
        $globMessages['mesDelete'] = "Сообщение было опубликовано";
        $result = 1;
    }
?>
```

После чего можно модифицировать запрос на извлечение данных и отображать только опубликованные данные, за исключением администратора, он должен видеть все сообщения.

```php
<?php
    $append = "";
    if(!isset($_SESSION["admin"])) {
        //обычный пользователь видит только опубликованные записи гостевой книги
        $append = "WHERE published = 1";
    }
    $queryResult = mysql_query("SELECT * FROM `gb_message` " . $append);
?>
```

Гостевая книга практически готова, осталось добавить возможность прикреплять файлы и тест Тьюринга.

2.12.5.4 Добавление изображения к сообщению (работа с файлами)

На форме с добавлением сообщения ранее мы добавили специальное поле ввода с типом file, которое позволяет загружать на сервер файлы любого типа. Так же мы добавили атрибут accept со значением imgae/*, используя этот атрибут, мы разрешаем пользователю выбирать только изображения, но этого не достаточно, так как исходный html-код легко редактируется, поэтому необходимо добавить соответствующую проверку на стороне сервера.

Загрузка файла на сервер делается абсолютно не сложно, после отправки информации на сервер, файл помещается во временную директорию сервера со временным именем, а после чего, используя функцию move_uploaded_file, копируем загруженный файл в нужную директорию. Вся информация о файле содержится в глобальном массиве $_FILES.

Перейдем к реализации, сначала мы должны модифицировать форму, чтобы была возможность передачи файлов для этого необходимо установить метод кодирования multipart/form-data, при отправке данных информация никак не кодируется.

```
<form method="POST" action="index.php" enctype="multipart/form-data" >
```

После отправки на сервер данных, массив $_FILES Будет содержать следующую информацию:

```
Array (
   [gb_mes_image] => Array (
       [name] => 09-04-2013 21;13;37.jpg
       [type] => image/jpeg
       [tmp_name] => /tmp/phpiF4Ikv
       [error] => 0
       [size] => 1024584
   )
)
```

Далее проверим, что файл успешно загрузился, и скопируем файл в папку files, а в таблицу «news» добавим путь до изображения.

```
<?php
$fileName = "";
if(isset($_FILES["gb_mes_image"]) && $_FILES["gb_mes_image"]["error"] == 0) {
    $fileName = "/files/".$_FILES["gb_mes_image"]["name"];
    // Проверяем загружен ли файл
    if(is_uploaded_file($_FILES["gb_mes_image"]["tmp_name"])) {
        // Если файл загружен успешно, перемещаем его из временной директории в
конечную
        move_uploaded_file($_FILES["gb_mes_image"]["tmp_name"], __DIR__.$fileName);
        $globMessages["fileUploaded"] = "Файл был успешно загружен";
        $result = 1;
    } else {
        $globMessages["fileUploaded"] = "Не возможно загрузить файл";
        $result = -1;
    }
}
//Если не возникло ранее ошибок, то добавляем сообщение
if($result != -1) {
    $result = addData($userName, $userMail, $userMessage, $fileName);
    if($result == 1) {
        //Отчищаем переменные в случае успешной записи
        $userName = "";
        $userMail = "";
        $userMessage = "";
    }
}
?>
```

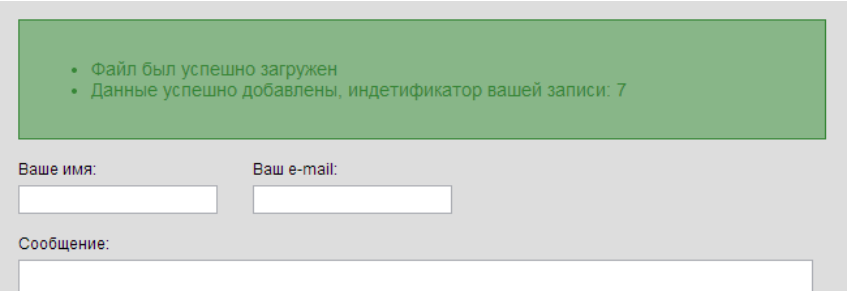

Рисунок 21 – Успешное добавление сообщения

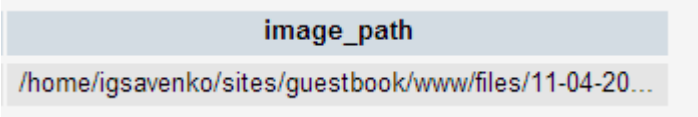

Рисунок 22 – Путь до изображения добавлен в БД

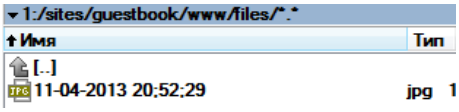

Рисунок 23 – Файл был успешно скопирован

Осталось только вывести изображение в тексте сообщения.

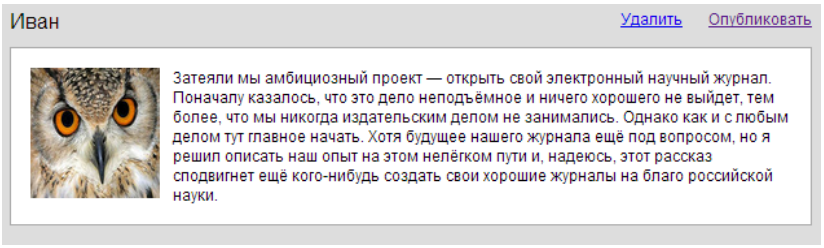

Рисунок 24 – Отображение изображения

Добавьте возможность в функцию checkData() проверку размера файла на стороне сервера, файл не должен превышать 500Кб, а также ограничение на расширение файла (пользователь может загружать только изображения), используя функцию pathinfo. Добавьте удаление файла после удаления сообщения (удаление файла осуществляется при помощи функции unlink).

Напоследок я оставил самое интересное - динамическое создание изображений.

2.12.5.5 Динамическое создание изображений

Динамическое создание изображений мы будем использовать при создании теста Тьюринга (CAPTCHA). Мы не будем придумывать хитрую и запутанное изображение, мы

попросим пользователя подсчитать простое математическое выражение. Как все это работает, мы сгенерируем 2 двухзначных числа, сформируем строку типа XX + XX = и наложим её на изображение. Подсчитаем результат и занесем его в сессию, это необходимо для того, чтобы можно было проверить результат. В случае если результат сошелся, то мы разрешаем добавлять новое сообщение.

Создадим php файл с именем «captcha.php», который будет генерировать изображение, важно отметить, что результатом выполнения сценария будет изображение, а не текстовый файл, поэтому мы воспользуемся функцией header, которая генерирует HTTP-ответ от сервера и сообщим клиенту, что ответом будет являться изображение в формате png.

```
<?php header ("Content-type: image/png"); ?>

<?php
    session_start();
    unset($_SESSION["captcha"]); //отчищаем переменную с суммой
    header ("Content-type: image/png");
    $img = imagecreate(150, 30);
    $background_color = imagecolorallocate($img, 255, 255, 255);
    $text_color = imagecolorallocate($img, 233, 14, 91);
    $fS = rand(10, 999); //генерируем первое слогаемое
    $sS = rand(10, 999); // генерируем второе слогаемое
    $summ = $fS + $sS; //суммируем
    $_SESSION["captcha"] = $summ; //пишем в сессию
    //размещаем текст на изображении
    imagestring($img, 5, 20, 5,  $fS." + ".$sS." =", $text_color);
    imagepng($img);
    imagedestroy($img);
?>
```

Вывод изображения очень прост: <img src="captcha.php">.Осталось добавить проверку в функцию checkData и гостевая книга полностью готова. Результат работы приведен на рисунке 25.

```
<?php
    //если переменные не пустые и значения не совпадают, значит тест не пройден
    if(empty($captcha) || empty($_SESSION["captcha"]) || $_SESSION["captcha"] !=
$captcha) {
        $errors["captcha"] = "CAPTCHA введена не корректно";
    }
?>
```

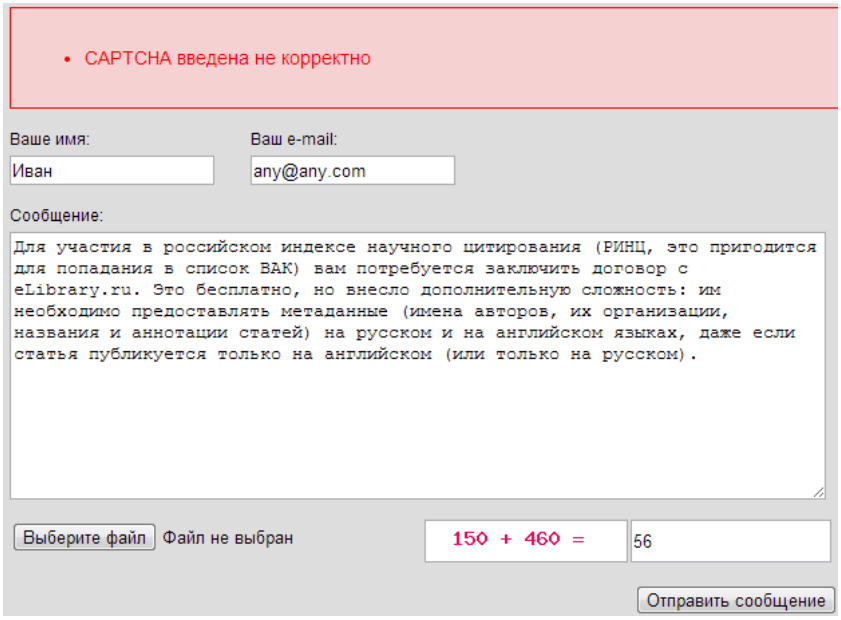

Рисунок 25 – Проверка введенного значения

Задание на лабораторную работу

1. Запустите гостевую книгу на локальном веб-сервер OpenServer с виртуальным хостом «project». Скопировав все исходные данные и создав пустую БД используя phpMyAdmin. Синхронизируйте модель БД с сервером через MySQL Workbench.
2. Изучите исходный код гостевой книги, внимательно прочитайте методическое пособие, выполните все задания (указанные в методическом пособие подчеркнутым шрифтом).
3. Добавьте возможность редактирования сообщения администратором.
4. Выполнить индивидуальное задание:

   Вариант 1. В гостевой книге на одной странице должны отображаться 5 последних сообщений. Под сообщениями реализовать постраничное отображение. Активная страница должна отображаться жирным шрифтом.

   Вариант 2. Добавить возможность пользователю указывать адрес его веб-сайта, по нажатию на ссылку страница должна отображаться в новом окне. Обязательно реализовать проверку адреса на корректность используя регулярные выражения.

   Вариант 3. Реализуйте следующий тест Тьюринга: Возьмите произвольное слово и удалите из него один любой символ (после перезагрузки страницы удаляется другой символ), выведите на изображение получившееся слово (на месте удаленного символа поставьте нижнее подчеркивание). Тест будет успешно пройден, если пользователь ввел недостающий символ.

   Вариант 4. Реализуйте следующий тест Тьюринга: Каждый раз при перезагрузке страницы возьмите произвольный цвет из массива. Закрасьте изображение этим цветом. Тест будет успешно пройден, если пользователь правильно определит отображаемый цвет. Ввод цвета осуществляется на русском языке (буквы могут быть как прописными, так и строчными).
5. Результат покажите преподавателю.
6. Интегрировать разработанную гостевую книгу в шаблон, который был сверстан в предыдущей лабораторной работе. Подготовьте отчет о проделанной работе.

ЗАКЛЮЧЕНИЕ

Задача данного методического пособия познакомить с основными возможностями языка PHP на примере разработки гостевой книги. В ходе разработки не применяется никаких сложных конструкций языка, только самое основное: базовые конструкции языка (функции, переменные, циклы, условия), сессии, работа с динамическими изображениями и загрузка файлов на сервер.

Чтобы код был красивее, а разработка более удобной, программисты использую Объектно-Ориентированный Подход к программированию, а также различные шаблоны проектирования, и PHP это позволяем в полном объеме. С использованием данного языка программирования, создано огромное количество систем управления сайтами и фреймворков, использование которых очень сильно облегчает разработку.

Приложение 1

(обязательное)

Функции PHP для работы с массивами

Мы уже рассмотрели некоторые функции, предназначенные для работы с массивами, например count, но их гораздо больше. Ниже приведены некоторые из самых простых функций, которые мы еще не рассматривали. Полный перечень функций вы найдете на сайте http://www.php.net.

```
<?php reset(массив) ?>
```

Принимает в качестве аргумента массив и переустанавливает указатель текущей позиции в начало массива. С помощью указателя (pointer) PHP запоминает текущий элемент массива при работе с функциями, которые могут перемещаться по массиву.

```
<?php array_push(массив, элементы) ?>
```

Добавляет один или больше элементов в конец существующего массива. Например, инструкция array_push($shapes, "камень", "бумага", "ножницы"); добавит в массив $shapes три указанных элемента.

```
<?php
   array_pop(массив)
?>
```

Удаляет и возвращает последний элемент массива. Например, инструкция $last_element=array_pop($shapes); удалит из массива $shapes последний элемент, присвоив его значение переменной $last_element.

```
<?php
   array_unshift(массив, элементы)
?>
```

Добавляет один или больше элементов в начало существующего массива. Например, инструкция array_unshift($shapes, "камень", "бумага", "ножницы"); добавит в начало массива $shapes три указанных элемента.

```
<?php
   array_shift(массив)
?>
```

Удаляет и возвращает первый элемент массива. Например, инструкция $first_element = array_shift($shapes); удалит первый элемент из массива $shapes, присвоив его значение переменной $first_element.

```
<?php
   array_merge(массив1, массив2)
?>
```

Объединяет два массива в один массив и возвращает новый массив. Например, инструкция $combined_array=array_merge($shapes, $sizes); объединит элементы двух массивов и присвоит новый массив переменной $combined_array.

```
<?php
   array_keys(массив)
?>
```

Возвращает массив, содержащий все ключи исходного массива. Например, инструкция $keys=array_keys($shapes); запишет в массив $keys значения ключей, такие как "Яблоко" и "Блокнот".

```
<?php
   array_values(массив)
?>
```

Возвращает массив с числовыми индексами, содержащий все значения элементов исходного массива. Например, инструкция $values=array_values($shapes); запишет в массив $values такие значения элементов, как "Шар" и "Прямоугольник".

```
<?php
   shuffle(массив)
?>
```

Переупорядочивает элементы массива случайным образом. В процессе переупорядочивания значения ключей исходного массива будут утеряны, поскольку на выходе получится массив с числовыми индексами.