Н.Б. Паршукова

ПРОГРАММИРОВАНИЕ С ИСПОЛЬЗОВАНИЕМ PHP И MYSQL В РАЗРАБОТКЕ ВЕБ-ПРИЛОЖЕНИЙ

Учебное пособие

Министерство просвещения Российской Федерации
Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Южно-Уральский государственный
гуманитарно-педагогический университет»

Н.Б. Паршукова

Программирование с использованием PHP и MySQL в разработке веб-приложений

Учебное пособие

Челябинск
2021

УДК 681.14 (021)
ББК 32.973.26-018.2я73
П 18

Паршукова, Н.Б. Программирование с использованием PHP и MySQL в разработке веб-приложений: учебное пособие / Н.Б. Паршукова. – Челябинск: Издательство Южно-Уральского государственного гуманитарно-педагогического университета, 2021. – 167 с.

ISBN 978-5-907409-81-1

Учебное пособие предназначено для обучения технологии создания веб-приложений на базе языка программирования PHP и системы управления базами данных MySQL. Уделяется внимание как теоретическим вопросам языка программирования PHP (синтаксис, алгоритмические конструкции, отличия от других языков веб-программирования) и MySQL (запросы, программное обеспечение, связь с PHP), так и практическому созданию веб-приложения с модульной архитектурой.

Пособие предназначено для бакалавров направления 09.03.02 Информационные системы и технологии, 44.03.01 Педагогическое образование (профиль Информатика) и 44.03.05 Педагогическое образование (профили Математика. Информатика, Физика. Информатика), может быть использовано для преподавания курсов «Программирование с использованием PHP и MySQL в разработке веб-приложений», «Разработка обучающего Web-ресурса средствами PHP и MySQL».

Рецензенты: Т.В. Карпета, канд. физ.-мат. наук, доцент
Н.А. Давыдова, канд. пед. наук, доцент

ISBN 978-5-907409-81-1

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Развитие Интернета, появление у людей множества устройств для доступа к нему усиливает тенденцию к разработке программного обеспечения, которое доступно через браузер. Уже длительное время язык программирования PHP является одним из лидеров языков для веб-разработки [3; 5; 12]. На базе этого языка создаются социальные сети, системы управления содержимым сайта (WordPress, Joomla, Битрикс и другие), корпоративные сайты, блоги, интернет-магазины и многие другие сервисы [11]. Хранение множества данных этих веб-приложений осуществляется с помощью систем управления базами данных (СУБД). MySQL – свободная реляционная система управления базами данных, которая в связке с PHP уже давно заслужила популярность у веб-разработчиков.

В данном учебном пособии рассматриваются теоретические вопросы программирования на PHP и работе с СУБД MySQL, приводятся примеры веб-сервисов, которые могут быть использованы, в том числе, для создания собственного тематического блога. Уделено внимание особенностям модульной архитектуры веб-приложений, вопросам безопасности хранения и обработки данных.

Представлены лабораторные работы для создания веб-приложения со списком статей, формой обратной связи для комментирования, авторизацией пользователя, фото галереей, тестированием. Описан подход к созданию интерфейса для панели управления веб-приложением (в качестве примера приведено управление отдельным модулем работы с фото галереей), который в дальнейшем может быть использован для разработки и панелей управления другими веб-сервисами.

Материал учебного пособия может быть полезен всем, кто знаком с программированием и хочет освоить разработку веб-приложений на языке PHP и СУБД MySQL, а также, студентам бакалавриата направления 09.03.02 Информационные системы и технологии (профиль «Информационные технологии в образовании»), 44.03.01 Педагогическое образование (профиль «Информатика»), 44.03.05 Педагогическое образование с двумя профилями подготовки (профиль «Математика. Информатика»).

ГЛАВА 1. ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ О ЯЗЫКЕ ВЕБ-ПРОГРАММИРОВАНИЯ PHP

1.1. Особенности серверного программирования на языке PHP

Язык PHP был разработан в 1994–1995 годах Расмусом Лэрдорфом, датским программистом. Изначально это был набор скриптов (сценариев) на языке программирования Perl под названием «PHP – Tools for Personal Home Page», который собирал статистику посещения веб-страницы. На сегодняшний день большинство страниц в сети Интернет снабжены подобными счетчиками. Скрипт имел открытый исходный код и в целом простую и понятную идею, которая понравилась другим разработчикам. Вскоре скрипты были переписаны на языке Си и были названы "Personal Homepages Tools". Новая реализация языка уже могла взаимодействовать с базами данных и с помощью него можно было создавать простые динамические веб-страницы, что послужило еще одним стимулом к развитию Интернета.

Надо отметить, что язык разметки гипертекста HTML, с помощью которого создаются веб-страницы, требует знания основных тегов, кропотливости в формировании веб-страницы. В середине 90-х годов прошлого века веб страницы в Интернете создавались программистами и большинство пользователей, если и имело доступ к сети, могли лишь просматривать страницы. Ценность идеи Р. Лэрдорфа состоит в том, что он создал достаточно простой инструмент для разработчиков, с помощью которого можно было создавать динамические веб-страницы, содержимое которых хранится в базе данных, а задачей PHP является извлечение этой информации из базы данных, обрамление тегами HTML для разметки веб-страницы. Таким образом, с помощью PHP можно было разработать удобный интерфейс для обычных людей, не знакомых с программированием, но владеющими пользовательскими навыками, чтобы уже пользователи могли формировать контент веб-страниц. Делалось это с помощью привычных нам полей форм: простое текстовое поле, многострочное поле, поле для ввода пароля, чекбоксы и др. Очень быстро PHP набирал популярность сре-

ди разработчиков в силу своей простоты, Си-ориентированного синтаксиса, все возрастающей поддержки многочисленных баз данных.

Создание языка PHP, при всех его несовершенствах, послужило взрывному развитию Интернета, достаточно сказать, что Facebook был написан на PHP и до сих пор содержит определенный функционал на PHP, хотя часть его выполняется на боле быстром C++. Панель редактирования Wikipedia реализована на PHP, а всем известно, что это универсальная интернет-энциклопедия, где может любой человек опубликовать свой контент, даже если он не знаком с программированием. Это лишь малая часть примеров. На PHP написано множество систем управления контентом: WordPress, Joomla, 1C Bitrix, Moodle, NetCat, Drupal и др., а с их помощью уже создаются современные сайты, порталы, информационные системы не только программистами, но и обычными пользователями. Такая система управления контентом, как WordPress, является одним из лидеров платформ для разработки сайтов во всем мире. Многие сайты образовательных учреждений реализованы с помощью богатого функционала всевозможных плагинов WordPress [11].

Среди всех языков программирования PHP уверенно входит в первую десятку, а среди языков веб-программирования является безусловным лидером. Поэтому этот простой и сравнительно легкий в освоении язык программирования может стать хорошим началом для тех, кто собирается разобраться в веб-технологиях и разработать свой веб-ресурс.

PHP – скриптовый язык, который непосредственно встраивается в HTML-код и выполняется сервером. Скриптовые языки (или языки сценариев) – это высокоуровневые языки, которые кратко описывают действия, выполняемые системой. По сути сценарий – это тоже программа, но чаще всего сценарии интерпретируются, а не компилируются. Это означает чуть более медленную работу программы, но обычный пользователь почти не почувствует разницы. Интерпретатор располагается на сервере и его задача –последовательно, команда за командой, считывать код сценария и, если в текущей инструкции нет ошибок, то выполнить код. Чаще всего задачей интерпретатора будет сгенерировать определенный HTML-код и отправить его обратно клиенту (браузеру). Таким образом, вся вычислительная нагрузка при работе интерпретатора PHP выполняется на удаленном сервере, а на компьютер пользователя присылается HTML-код. Если в какой-то момент интерпретатор обнаружит ошибку в коде, то он может вывести часть HTML-кода до ошибки и прекратить свою работу,

встретив ее. Таким образом, веб-разработчик может столкнуться с ситуацией, когда часть страницы построилась, а часть – нет, и в браузере появилось сообщение об ошибке. Компилируемые программы (C, C++, C#, Pascal, Java и др.) попросту не запустятся, если при компиляции обнаружена ошибка.

Основное отличие PHP от CGI-скриптов, написанных на других языках, типа Perl или C++, – это то, что в CGI-программах необходимо прописывать выводимый HTML-код, а используя PHP, он встраивается в готовую HTML-страницу при помощи открывающего и закрывающего тегов

`<?  …  ?>` или `<?php … ?>`

Отличие PHP от JavaScript состоит в том, что PHP-скрипт выполняется на сервере, а клиенту передается результат работы, тогда как JavaScript-код полностью передается на клиентскую машину (в браузер) и только там выполняется.

В PHP включена поддержка многих баз данных, что делает написание веб-приложений с использованием БД достаточно простым: MySQL, Adabas D, InterBase, Solid, dBase, mSQL, Sybase, PostgreSQL, Empress, Velocis, File Pro, Oracle, Unixdbm, Informix.

CGI (Common Gateway Interface – общий интерфейс шлюзов) является стандартом, который предназначен для создания серверных приложений, работающих по протоколу HTTP. Такие приложения (их называют шлюзами или CGI-программами) запускаются сервером в режиме реального времени.

Сервер передает запросы пользователя CGI-программе, которая их обрабатывает и возвращает результат своей работы на экран пользователя. Таким образом, посетитель получает динамическую информацию, которая может изменяться в результате влияния различных факторов. Сам шлюз (скрипт CGI) может быть написан на различных языках программирования – Си/C++, Fortran, Perl, TCL, UNIX Shell, Visual Basic, Python и др.

При обращении пользователя к статической странице на сервере происходит следующее (рисунок 1): браузер посылает на сервер запрос на получение содержимого файла, например news.html. При этом на сервере в файловой системе действительно существует файл news.html, и сервер попросту отправляет содержимое этого файла назад браузеру.

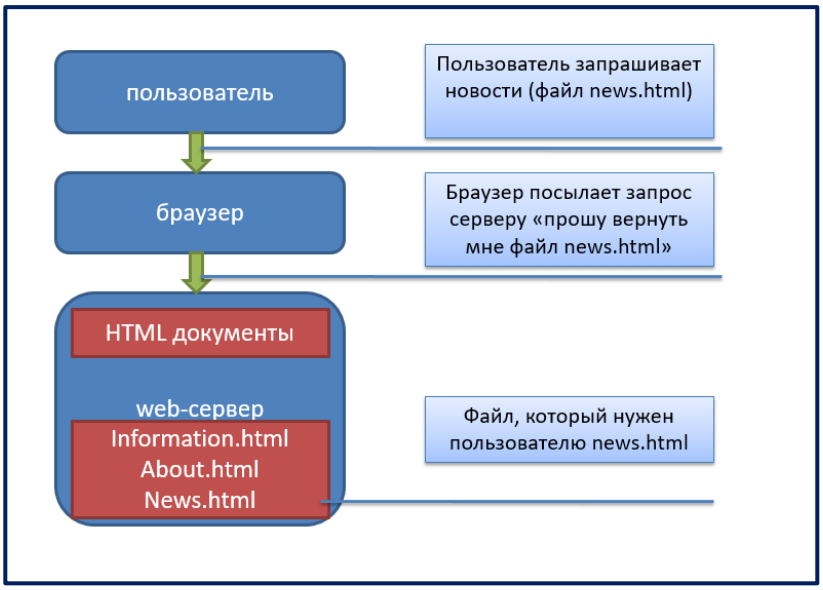


Рис. 1. Взаимодействие между сервером и клиентом. Обработка запроса

При обращении же к динамической PHP-странице, например program.php, происходит другой порядок действий. В физическом смысле program.php содержит код программы, которая формирует динамический HTML-код. Это значит, что при определенных обстоятельствах этот код будет генерировать разные веб-страницы. Например, program.php может извлекать из базы данных все статьи и сортировать их определенным образом. А базу данных со статьями могут постепенно изменять. Таким образом динамическая страница в физическом смысле не содержится на сервере (в отличие от статичной), а генерируется PHP-программой (рисунок 2).

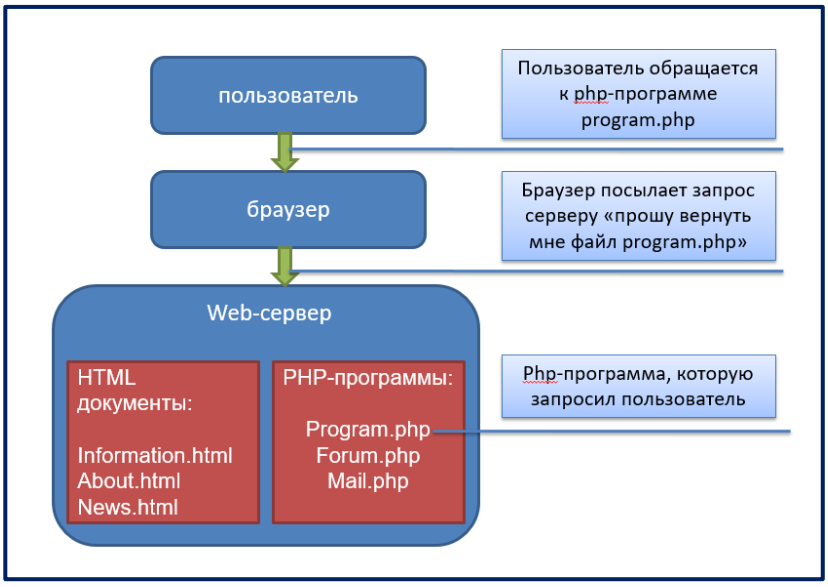


Рис. 2. Взаимодействие между сервером и клиентом. Обращение пользователя к php-программе

Сервер находит запрашиваемую PHP-программу, организует ее выполнение (с помощью интерпретатора php.exe). После того как программа выполнена, получается обычный статичный HTML-файл, в котором можно увидеть только результат.

Сервер возвращает браузеру результат работы PHP-программы, причем пользователь видит его как обычный HTML-код и даже не подозревает, что это была программа.

Все ошибки в PHP выдаются в окно браузера по умолчанию (чего нельзя сказать о CGI, где все ошибки записываются в лог-файл), и найти их при внимательном поиске и определенном опыте не составит труда. Тем более, что интерпретатор подскажет номер строки, в которой произошла ошибка.

При написании программы на PHP необходимо пользоваться разделением инструкций – в PHP инструкции разделяются так же, как в C или Pascal, – точкой с запятой.

Необходимо заметить, что закрывающий тег ?> также подразумевает завершение инструкции, другими словами, если после инструкции следует закрывающий тег ?>, то можно не использовать точку с запятой в конце инструкции. Например,

```
<?php
echo( "Welcome to PHP");
?>
```

То же самое, что и

```
<?php echo ("welcome to PHP") ?>
```

Для разработки веб-проекта на языке PHP понадобится особое программное обеспечение. Во-первых, веб-разработчик должен иметь на компьютере надежный и удобный браузер для отображения страниц. Наиболее популярными, быстрыми и удобными для разработчиков являются браузеры Google Chrome, Яндекс.Браузер, Opera, Edge и др. Иногда полезно иметь несколько установленных на компьютере браузеров, так как порой приходится тестировать верстку сайта, которая может по-разному отрисовываться в браузерах.

Для того чтобы программа на PHP была выполнена, необходим интерпретатор PHP, который работает на веб-сервере. Веб-сервер – это программа, которая способна воспринимать HTTP запросы от клиента (браузера) и отдавать обратно HTTP ответы (с HTML-кодом, изображениями, файлами и др. контентом). Самыми доступными веб-серверами являются бесплатный Apache и входящий в состав операционной системы

Windows веб-сервер IIS (Internet Information Server). В данном курсе будет использоваться веб-сервер Apache в силу распространенности, простой настройки и свободного распространения.

При работе с базами данных также понадобится сервер баз данных. В нашем случае работа будет осуществляться с использованием системы управления базами данных MySQL, поэтому именно сервер MySQL нам также понадобится для работы. Обычно для одного веб-проекта создается одна база данных, которая хранится на удаленном сервере, и пользователи со своих устройств могут получать доступ к этой базе данных через сайт, через интерфейс сайта формировать запросы к этой базе данных. Создание и изменение баз данных выполняется с помощью графической утилиты phpMyAdmin, которая является частью веб-сервера.

Все вышеперечисленные компоненты для разработки сайта – веб-сервер Apache, MySQL, интерпретатор PHP – входят в набор дистрибутивов, который обозначается аббревиатурой LAMP (L – Linux, A – Apache, M – MySQL, P – PHP). Набор указанных дистрибутивов входит в популярный набор Denwer Дмитрия Котерова. Для наших занятий мы будем использовать Denwer.

Для быстрого написания кода можно использовать текстовый редактор, такой как Notepad++, Atom, Sublime Text 3 и др. При выборе текстового редактора для написания кода следует обратить внимание на подсветку синтаксиса, выделение ошибок, работу с множеством файлов, быстрое перекодирование страниц в различные кодировки и др. В наших лабораторных работах будет использоваться Notepad++.

Далее, опишем процедуру развертывания веб-сервера Apache на операционную систему Windows 10. Если вы планируете работать на веб-сервере IIS от Microsoft, инструкцию по установке можно посмотреть в учебном пособии [3, с. 8]. Установка PHP под Linux представлена в учебном пособии [19, с. 21].

Также необходимо позаботиться о наличии свободного места на жестком диске для работы сервера (300-500 Мб).

Для начала работы с PHP требуется сделать следующее.

1. Скачать уже установленный пакет denwer по ссылке https://cloud.mail.ru/public/4UCB/yTo1fBxY7 и распаковать архив на Рабочий стол. Если вы действуете самостоятельно, то можно выполнить установку Denwer по инструкции с сайта http://www.denwer.ru или скачать и установить OpenServer. В данном учебном пособии все работы проводятся на уже установленной сборке Denwer.

2. Выполнить запуск сервера. Для этого в папке **WebServers** (у вас может быть немного другое название) в папке **denwer** найдите три .exe файла: **Run.exe**, **Stop.exe, Restart.exe** (рисунок 3). Как нетрудно догадаться, с их помощью происходит запуск сервера, остановка и перезапуск. Для запуска сервера нужно запустить программу **Run.exe**.

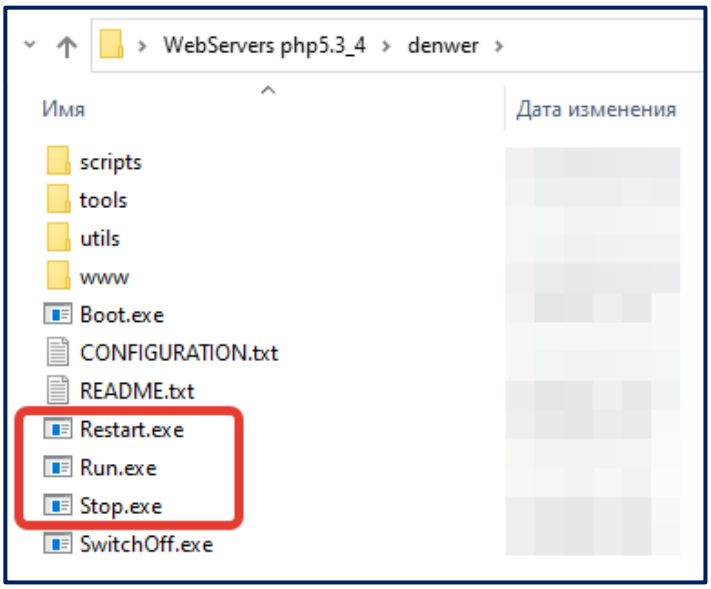


Рис. 3. Запуск, остановка и перезапуск сервера

При этом иногда операционная система может запросить у вас логин и пароль администратора для изменения файла hosts, но в этом случает не нужно вводить никакого пароля. Появится некритическая ошибка, но она не повлияет на дальнейшую работу.

Если вы все сделали правильно, то в области рядом с часами на своем компьютере увидите ярлыки (рисунок 4).

Рис. 4. Запуск, остановка и перезапуск сервера

3. Далее необходимо создать папку, где будут храниться все файлы сайта. Для этого в папке **WebServers/home/localhost/www** создайте папку, например **program** (следует обратить внимание, что все имена папок, файлов при веб-разработке следует именовать строчными буквами, без пробелов на латиннице), рисунок 5.

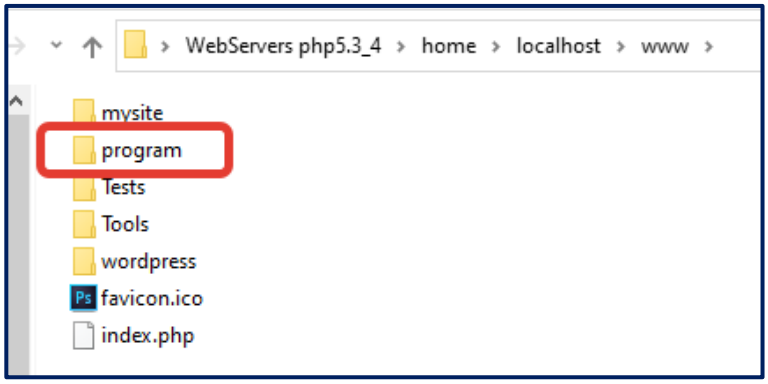


Рис. 5. Запуск, остановка и перезапуск сервера

4. В папке **program** создаются и располагаются файлы сайта и различные ресурсы (изображения, файлы стилей и др.). Традиционно стартовый файл сайта называют **index.php**, и он веб-сервером Apache воспринимается стартовым по умолчанию. Это значит, что дополнительно в адресной строке прописывать название файла не нужно. Создадим файл index.php с помощью Notepad++ со следующим кодом

```
<?php echo "Это мой первая страница!"?>
```

При работе с кириллицей требуется смена кодировки файла. Сделать это можно в Notepad++ с помощью пункта меню **Кодировки/Преобразовать в ANSI (**или **Encoding / Convert to ANSI).**

5. После этого можно запустить браузер и ввести адрес http://localhost/program. Здесь уже не нужно указывать папку www и файл **index.php** (см. рисунок 6).

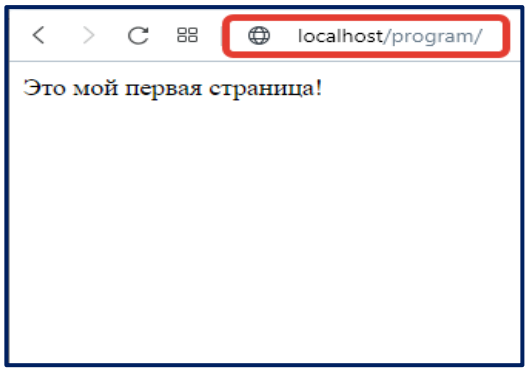


Рис. 6. Запуск, остановка и перезапуск сервера

Мы рассмотрели особенности языка веб-программирования PHP, необходимое для веб-разработки программное обеспечение, а также последовательность запуска веб-сервера и создание простейшей страницы.

Также, следует обратиться к официальной документации по PHP на русском языке https://www.php.net/manual/ru/ при появлении вопросов по неописанным в этом учебном пособии функциям, операторам и синтаксису.

1.2. Синтаксис языка PHP

В PHP, впрочем, как и в C и PASCAL, допустимы две разновидности комментариев. Комментарии традиционного стиля начинаются знаком /* и заканчиваются знаком */.

Например,

```
/* это комментарий в одну строку*/
/* этот комментарий занимает
несколько строк*/
// этот комментарий занимает
// несколько строк
```

В качестве разделителя инструкций в PHP, как и во многих других языках программирования, используется точка с запятой «;».

В PHP определены следующие типы данных:

- Integer – целое,
- double – число с дробной частью,
- string – строковая переменная,
- array – массив,
- object – объектная переменная,
- boolean – булев (логический).

Особенностью синтаксиса PHP является то, что все переменные имеют имя, которые начинаются со знака **$,** например, **$value = 5**. При этом переменную очень легко отличить от остального кода. Однозначная идентификация переменной позволила создателям PHP дать возможность программистам использовать переменные непосредственно внутри строк. При этом явно указывать тип переменной в PHP не нужно. Интерпретатор самостоятельно определяет тип переменной в зависимости от содержимого.

Примеры определения различных переменных на PHP:

```
$str      = 'строка';
$str2     = "Это тоже строка";
$number   = 3;
$pi       = 3.1415926;
```

В PHP различаются строки, заключенные в одинарные и двойные кавычки. Подобное замещение имен в переменных их значениями производится только в строках, заключенных в двойные кавычки. Это очень важно помнить, дабы избежать ошибок в будущем.

```
$name = 'Maria';
$age = 20;
echo "$name is $age years old";
```

Переменная характеризуется именем, типом и значением. Имя переменной может быть любым и включать в себя цифры, буквы английского алфавита, а также разрешенные символы (например, символ подчеркивания или тире).

Необходимо заметить, что имя переменной чувствительно к регистру.

```
$first = 'Sasha';
$First = 'Tatiana';
echo "$first, $First"; // данная функция выведет
                        //на экран "Sasha, Tatiana"
```

Некоторые символы при написании кода на PHP являются служебными. Если нужно использовать определенный символ с его особым значением, то необходимо экранировать данный символ. Так, например, часто приходится внутри строки, обрамленной кавычками, использовать еще кавычки. В этом случае PHP будет выдавать ошибку. Например, при выводе строки

```
<?php echo "Это "строка" в кавычках";?>
```

получим ошибку. Чтобы правильно вывести такую строку, нужно перед внутренними кавычками поставить знак обратного слеша «\»:

```
<?php echo "Это \"строка\" в кавычках";?>
```

В таблице 1 представлены другие символы, которые нужно выводить с обратным слешем, чтобы получить желаемый символ.

Таблица 1

Символы, кодируемые с использованием обратного слеша

Последовательность	**Значение**
\n	Перевод строки (LF или OxOA в кодировке ASCII)
\r	Возврат каретки (CR или OxOD в кодировке ASCII)
\\t	Горизонтальная табуляция (HT или 0x09 в кодировке ASCII)
\\	Обратный слеш

Последовательность	Значение
\$	Знак доллара
\"	Двойная кавычка
\[0-7]{1,3}	Последовательность символов, соответствующая регулярному выражению, является символом в восьмеричной нотации
\x[0-9A-Fa-f]{1,2}	Последовательность символов, соответствующая регулярному выражению, является символом в шестнадцатеричной нотации

Рассмотрим вопросы преобразования строк. Если строка оценивается как числовое значение, то результирующие значение и тип будут определяться следующим образом. Строка оценивается как число двойной точности, если она содержит любой из символов «.», «e» или «E». В противном случае она будет оценена как целое число. Далее значение определяется начальной частью строки. Если строка начинается с допустимых числовых данных, они будут использоваться в качестве значения. В противном случае значение будет равно нулю. Экспонента – это «e» или «E», за которой следуют одна или несколько цифр. Если первое выражение является строкой, тип переменной зависит от второго выражения.

Пример преобразования строк:

```
$x = 1 + "10.5";          //$x имеет двойную точность (11.5)
$x = 1 + "-1.3e3";        //$x имеет двойную точность (-1299)
$x = 1 + "bob-1.3e3";     // $x целое (1)
$x = 1 + "bob3";          // $x целое (1)
$x = 1 + "10 Small Pigs";// $x целое (11)
$x = 1 + "10 Little Piggies"; // $x целое (11)
$x = "10.0 pigs " + 1;    // $x целое (11)
$x = "10.0 pigs " + 1.0; //  $x имеет двойную точность (11)
```

При работе со строками часто приходится выполнять операцию слияния строк или конкатенацию. В PHP эта операция обозначается символом точки «.». Приведем пример различных использований конкатенации.

```
<?php
     //определение строки
     $str = "Это строка";
     //добавление к ней другой строки
     $str = $str . " с дополнительным текстом";
     //еще один короткий способ добавления строки к строке
     $str .= " и перевод строки в конце. \n";
```

```
        //эта строка закончится текстом '<p>Число 3</p>'
        $number = 3;
        $str = "<p>Число $number</p>";
        //эта    строка    закончится    текстом    '<p>Число
$number</p>'
        $number = 5;
        $str = '<p>Число $number</p>';
        $str = "Программирование";
        //получение первого символа 'П'
        $first = $str[0];
        //получение последнего символа 'е'
        $last  = $str[strlen($str)-1];
    ?>
```

Строки являются упорядоченными последовательностями символов, поэтому к каждому символу строки можно обратиться по его порядковому номеру. В PHP нумерация строк начинается с 0. Чтобы вычислить всю длину строки $str (количество символов в строке), используется функция strlen($str).

Массивы – один из важных структурных типов данных в php. Они широко применяются при взаимодействиях с базами данных. По своей сути массив – это ряд переменных, упорядоченных по имени и имеющих различный индекс (ключ).

```
$m[0] = "компьютер";
$m[1] = "Интернет";
$m[2] = "модем";
$m[3] = "монитор";
$i = 0;
while ($i < count($m)) {
     echo $m[$i]."<br>";
     $i++;
}
```

Различают одномерные массивы и многомерные массивы. Одномерный массив состоит из одной строки.

PHP поддерживает как скалярные (в качестве ключей используются порядковые номера 0, 1, 2...), так и ассоциативные массивы (в качестве ключей используется строка "a", "Ь", "c"...). Между ними, практически, нет никакой разницы.

Создать массив можно с помощью функций **array()** или просто определить значения элементов массива. С помощью функции **print_r** можно вывести весь массив целиком.

```
<?php
     $m = array();
     $m[] = 1;
     $m[] = 2;
     $m[4] = 4;
     print_r($m);
?>
```

Код, указанный выше, можно написать более компактно, например, так:

```
<?php
     $m = array(0=>1,1=>2,4=>4);
     print_r($m);
?>
```

Для работы с массивами служит специальный цикл foreach, который мы рассмотрим ниже. Заранее можно сказать, что с помощью массива foreach можно обратиться последовательно к каждому элементу массива и его ключу и произвести с ними какие-то действия.

Если использовать ассоциативные массивы, то в качестве ключа будет выступать строка, как, например, здесь, массив $fruit хранит информацию о фрукте яблоке (его цвет, вкус, форма, название и числовая характеристика).

```
$fruit["color"] = "red";
$fruit["taste"] = "sweet";
$fruit["shape"] = "round";
$fruit["name"] = "apple";
$fruit[3] = 4;
```

Или можно так:

```
$fruit = array(
"color" => "red",
"taste" => "sweet",
"shape" => "round",
"name" => "apple",
3 => 4
) ;
```

Многомерный массив – это массив, каждый элемент которого в свою очередь тоже является массивом. В данном примере в массиве **$fruit** описаны три фрукта и выведен вкус апельсина. Какой он?

```
<?php
     $fruit = array(
          "apple"=>array(
                    "color"=>"red", "taste"=>"sweet",
"shape"=>"round"),
          "orange"=>array(
                    "color"=>"red", "taste"=>"tart",
"shape"=>"round"),
          "banana"=>array(
                    "color"=>"yellow", "taste"=>"pasty",
"shape"=>"banana-shaped" )
     );
     echo $fruit["orange"]["taste"];
?>
```

В любом языке программирования, в том числе и в PHP, используются операции для записи некоторых действий. Например, из математики широко известны арифметические операции – сумма, разность, умножение, деление. Сюда можно отнести также нахождение целой части и остатка от деления двух чисел. Таким образом, операции являются частью синтаксических конструкций, которые позволяют сделать определенные действия с переменными.

Унарные операции, или одноместные операции, подразумевают, что число операндов (переменных), над которыми операция производит свое действие, равно единице. К ним относятся операция присваивания (=), инкремент (++), декремент (--) и др.

```
$number = 5;
$number++; // то же, что $number = $number +1 или
                //$number += 1
--$number; // то же, что $number = $number -1 или
                //$number -= 1
```

Префиксный инкремент **увеличивает** $a на единицу, затем **возвращает** значение $a. Запись **++$a**.

Постфиксный инкремент **возвращает** значение *$a*, затем **увеличивает** *$a* на единицу. Запись **$a++.**

Префиксный декремент **уменьшает** *$a* на единицу, затем **возвращает** значение *$a*. Запись **--$a**.

Постфиксный декремент **возвращает** значение *$a*, затем **уменьшает** *$a* на единицу. Запись **$a--**.

Рассмотрим на примере как работают инкременты и декременты в различных ситуациях.

```
<?php
     echo "<h3>Постфиксный инкремент</h3>";
     $a = 5;
     echo "Должно быть 5: " . $a++ . "<br />\n";
     echo "Должно быть 6: " . $a . "<br />\n";
     echo "<h3>Префиксный инкремент</h3>";
     $a = 5;
     echo "Должно быть 6: " . ++$a . "<br />\n";
     echo "Должно быть 6: " . $a . "<br />\n";
     echo "<h3>Постфиксный декремент</h3>";
     $a = 5;
     echo "Должно быть 5: " . $a-- . "<br />\n";
     echo "Должно быть 4: " . $a . "<br />\n";
     echo "<h3>Префиксный декремент</h3>";
     $a = 5;
     echo "Должно быть 4: " . --$a . "<br />\n";
     echo "Должно быть 4: " . $a . "<br />\n";
?>
```

К унарным операциям можно отнести и логическую операцию «отрицание», которая обозначается символом восклицательного знака «!». Например, с помощью оператора **var_dump** можно посмотреть тип и значение логической переменной (нетрудно догадаться, что ее значение будет **false**):

```
<?php
     $a = true;
     var_dump(!$a);
?>
```

Бинарные операции, или «двухместные», означают, что операции этого рода производят свои действия над двумя операндами. Наиболее часто употребляемые двухместные операции – это +, -, / и *. Также к двуместным операциям можно отнести логические операции (равно, не равно, меньше или равно, меньше, больше или равно, больше, логические И, логическое ИЛИ и др.).

В таблице 2 представлены бинарные операции.

Таблица 2

Основные бинарные операции в PHP

Обозначение	**Название**	**Синтаксис**
*	Умножение	выражение * выражение
+	Сложение	выражение + выражение
–	Вычитание	выражение – выражение
/	Деление	выражение / выражение
%	Модуль (остаток деления)	выражение % выражение
==	Равно	выражение == выражение
!=	Не равно	выражение != выражение
===	Идентично	выражение === выражение
!==	Не идентично	выражение !== выражение
>	Больше	выражение > выражение
>=	Больше или равно	выражение >= выражение
<	Меньше	выражение < выражение
<	Меньше или равно	выражение <= выражение
||	Логическое Или	выражение ||выражение
&&	Логическое И	выражение && выражение

Кроме обычного сравнения ==, есть сравнение тождественности ===, возвращающее **true**, если операнды равны по значению и имеют один тип.

Сравнение тождественности «не равно по типу и значению» записывается как !==, в отличие от обычного отношения неравенства, которое обозначается != или <>.

Более подробную информацию по работе с операторами PHP можно получить из [9; 18].

При разработке веб-сайта часто приходится вставлять код одного файла в другой. Например, вы сделали файл, который выводит меню или подключается к базе данных. Теперь точно такой же код должен находиться на всех страницах вашего сайта. Если в процессе работы разработчику необходимо внести в код какие-либо изменения (добавить новый пункт в меню или изменить данные для подключения к базе), то править этот код во множестве файлов представлялось бы рутинной и неэффективной задачей. Операторы включения кода позволяют создавать один исходный файл и затем подключать его к каким нужно файлам.

Рассмотрим их. Оператор **include** позволяет включать код, содержащийся в указанном файле, и выполнять его столько раз, сколько программа встречает этот оператор. Включение может производиться любым из перечисленных способов:

```
include 'имя_файла';
include $file_name;
include ("имя_файла");
```

Пусть имеется файл **data.php**, которые содержит некоторые сведения

```
<?php
     $users = array("Иванов И.И.", "Сидоров В.А.", "Ерофеева М.Ю.");
     $events = array("Конференция по информационным технологиям", "Семинар по математике", "Мастер-класс по искусственному интеллекту");
?>
```

Файл **data.php** подключается к файлу **index.php**

```
<?php
include ("data.php");
echo "Здравствуйте, $users[1]!<br>";
// выведет "Здравствуйте, Иванов И.И.!"
echo "Вы приглашены на мероприятие \"$events[2]\"";
// выведет, например, "Вы приглашены на мероприятие "Семинар по математике""
?>
```

Использование оператора **include** эквивалентно простой вставке содержательной части файла **data.php** в код программы **index.php**. Может быть, тогда можно было в **data.php** записать простой текст без всяких тегов, указывающих на то, что это php-код? Нельзя! Дело в том, что в момент вставки файла происходит переключение из режима обработки PHP в режим HTML. Поэтому код внутри включаемого файла, который нужно обработать как PHP-скрипт, должен быть заключен в соответствующие теги.

Функция **include_once** работает идентично выражению **include**, с той лишь разницей, что если код из файла уже один раз был включён, он не будет включён и выполнен повторно и вернёт **true**. Как видно из имени, он включит файл только один раз.

Функция **require** также позволяет включать в программу и исполнять какой-либо файл. Основное отличие **require** и **include** заключается в том, как они реагируют на возникновение ошибки: **include** выдает предупреждение, и работа скрипта продолжается. Ошибка в **require** вызывает фатальную ошибку работы скрипта и прекращает его выполнение.

Функция **require_once** обеспечит добавление кода к сценарию только один раз и поможет избежать столкновений со значениями переменных или именами функций.

Может возникнуть резонный вопрос: в каких случаях использовать ту или иную функцию, если все они включат исходный код. Наиболее строго функцией является **require** – если вы сделали ошибку, то на странице не будет выведено ничего. Это лучший вариант в том случае, если сайт размещен в сети Интернет – так злоумышленники не увидят сообщение об ошибке, что может дать дополнительную информацию к взлому сайта. Но если вы ведете разработку на локальном хостинге и вам нужно понимать, где вы совершили ошибку, то **include** или **include_once** будет предпочтительнее в этом случае. **Include_once** и **require_once** используются в тех случаях, если подключаемые файлы содержат определения функций. В этих случаях функция будет описана только один раз и это не приведет к ошибке повторного описания функций.

Представленные функции повышают удобство сопровождения сайта и улучшают его масштабируемость, т.к. позволяют менять код всего лишь одного файла, а остальные страницы сайта автоматически подтягивают его из исходного.

1.3. Алгоритмические конструкции в PHP

Алгоритмы состоят из трех базовых алгоритмических конструкций – следований, ветвлений и циклов. Следование – это последовательное выполнение инструкций в программе, где каждая отделена от другой точкой с запятой. В некоторых приведенных выше примерах были представлены последовательные алгоритмы.

В тех случаях, когда программа должна выполнять те или иные действия в зависимости от логических условий, значений переменных, результатов выполнения функций, используется алгоритмическая конструкция **ветвление**. В PHP к условным операторам относят оператор **if** и оператор **switch**. Рассмотрим их работу.

Оператор **if** делится на две формы – неполная и полная. Неполная форма условного оператора на PHP выглядит следующим образом:

```
If (условие) {
     Оператор_1;
     Оператор_2;
     …
     Оператор_N;
}
```

Оператор **if** (если) проверяет выполнимость условия (в результате должно получиться либо логическое true – истина, либо логическое false – ложь). В случае, если условие имеет значение true, выполняются действия операторов 1…n. Если же условие имеет значение false, то программа переходит к выполнению оператора, следующего за оператором if.

Полная форма условного оператора выглядит следующим образом:

```
If (условие) {
     Оператор_ветви_Если_то_1;
     Оператор_ветви_Если_то_2;
     …
     Оператор_ветви_Если_то_N;
}
else
{
     Оператор_ветви_Если_Иначе_1;
     Оператор_ветви_Если_Иначе_2;
     …
     Оператор_ветви_Если_Иначе_N;
}
```

Оператор **if** (если) проверяет выполнимость условия (в результате должно получиться либо логическое true – истина, либо логическое false – ложь). В случае, если условие имеет значение true, выполняются действия операторов_ветви_Если_то 1…n. Если же условие имеет значение **false**, то выполняются действия операторов_ветви_если_иначе 1…n.

Существует альтернативный синтаксис условного оператора, который удобен в том случае, если в программном коде предполагается несколько вложенных управляющих конструкций (чтобы разработчику не пришлось путаться в закрывающихся скобках множества вложенных конструкций):

```
If (условие):
Оператор_1;
Оператор_2;
…
Оператор_N;
Endif;
```

Здесь основной формой альтернативного синтаксиса является изменение открывающей фигурной скобки на двоеточие (:), а закрывающей скобки на Endif.

В качестве условия в алгоритмической конструкции **if** выступает логическое выражение, которое может состоять из логических операций и операций сравнения. К логическим операциям относят логическое И (&&), логическое ИЛИ (||), логическое НЕ (!) (таблица 2). К операциям сравнения относят восемь операций: равно, не равно, больше, больше или равно, меньше, меньше или равно, идентично, не идентично (таблица 2).

Если в алгоритме разработчику нужна альтернатива из более чем двух вариантов, то целесообразнее применить алгоритмическую конструкцию «оператор варианта» **switch** (переключатель). Его синтаксис:

```
switch (выражение или переменная) {
     case значение1: блок_действий1 break;
     case значение2: блок_действий2 break; ...
     default: блок_действий_по_умолчанию
          }
```

Switch проверяет значение переменной или выражения, сопоставляет вычисленное значение с одним из вариантов case. Если вариант выбора найден, то выполняется соответствующий блок действий. Если не найден вариант, то выполняется ветка default (по умолчанию). Иногда ветка default может отсутствовать, тогда при отсутствии нужного варианта управление переходит на оператор, следующий после **switch**.

Третий вид алгоритмических конструкций – **циклические операторы** – предназначены для программирования повторяющихся действий. Общая структура всех циклов предполагает действия, которые будут выполняться **n**-ое количество раз и условие прекращения цикла. Важно обязательно продумывать логическое условие и способ его выполнения и невыполнения, чтобы это не приводило к зацикливанию. В PHP используется четыре вида циклов. Рассмотрим их синтаксис и возможности использования.

Цикл с предусловием **while** предполагает сначала проверку условия. В случае если условие истинно, происходит выполнение действий в цикле.

```
while (выражение)
{
     блок_выполнения
}
```

Альтернативный синтаксис цикла с предусловием

```
while (выражение):
     блок_выполнения
endwhile;
```

Пример работы цикла с предусловием

```
<?
//эта программа напечатает все четные цифры
     $i = 1;
     while ($i < 10) {
          if ($i % 2 == 0) print $i;
          // печатаем цифру, если она четная
          $i++; // и увеличиваем $i на единицу
}
?>
```

Здесь в теле цикла включен оператор постфиксного инкремента **$i++**, который увеличивает переменную **$i** на единицу, что приводит к изменению условия, и тем самым рано или поздно цикл завершит свою работу.

Цикл с постусловием применяется, если гарантированно требуется совершить хотя бы одно действие прежде, чем выполнится проверка условия. Синтаксис цикла с постусловием:

```
do
     {
          блок_выполнения
     }
while (выражение);
```

Пример работы цикла с постусловием, который выводит подряд все цифры от 0 до 10:

```
<?
     $num = 0;
     Print "счетчик цикла do: ";
     do
          print "$num";
     while ($num++ <10);
?>
```

Если заранее известно сколько раз должен выполниться цикл, то целесообразнее применять цикл с параметром for. У него можно задать начальное значение, условие продолжения цикла и выражение для управления циклом (достижение условия выхода):

```
For (начальное_выражение; условие; выраже-
ние_управления_циклом)
```

```
{
     блок_выполнения
}
```

В некоторых случаях можно опустить начальное значение, условие продолжения цикла и выражение для управления циклом. Их можно заменить дополнительными проверками внутри цикла. Этот механизм позволяет быть PHP очень гибким. Но если вы не привыкли к такому синтаксису или боитесь ошибиться, то применяете классическую запись.

Пример работы цикла с параметром, который выводит последовательно цифры от 1 до 10:

```
for ($i = 1; $i <= 10; $i++)
{
     print $i;
}
```

Того же самого можно достичь при использовании проверки выполнения тела цикла не в заголовке, а в теле цикла:

```
for ($i = 1;;$i++){
     if ($i > 10) { break;}
     print $i;
}
```

Цикл «для каждого» **foreach** применяется только к массивам и объектам. При работе с упорядоченными элементами важно обращаться как к самим значениям, так и к их ключам (индексам). Именно поэтому был разработан данный вид цикла:

```
Foreach (array_expression as $value) {
     блок_выполнения
}
```

Здесь в качестве **array_expression** выступает переменная, являющаяся или массивом, или объектом.

При обращении к ключам и значениям элементов используется полная форма цикла **foreach**:

```
Foreach (array_expression as $key => $value) {
     блок_выполнения
}
```

Рассмотрим пример работы цикла **foreach**, который выводит в первом случае только названия стран, а во втором – название страны и код, который она имеет.

```
<? $countries = array("RU"=>"Россия",
                      "CN"=>"Китай",
                      "FR"=>"Франция",
                      "DE"=>"Германия");
foreach ($countries as $country){
```

```
          echo "Страна: ".$country."<br>";
     }
     foreach ($countries as $code => $country){
          echo "Страна: ".$country." имеет код " . $code
."<br>";
     }
     ?>
```

В PHP существуют два оператора, которые позволяют принудительно прервать действие цикла, не прибегая к проверке условия цикла – операторы **break** и **continue**.

Оператор **break** (разбить, прервать) прерывает исполнение ближайшего внешнего оператора **while**, **do-while**, **for** или **switch**. Управление передается следующему за прерванным оператору. **Break** принимает необязательный числовой аргумент, который сообщает ему, выполнение какого количества вложенных структур необходимо прервать. Значение по умолчанию 1, только ближайшая структура будет прервана.

```
<?php
$arr = array('один', 'два', 'три', 'четыре', 'стоп', 'пять');
foreach ($arr as $val) {
     if ($val == 'стоп') {
       break;    /* Тут можно было написать 'break 1;'. */
         }
     echo "$val<br />\n";
}
?>
```

Оператор **continue** (продолжай) прекращает текущую итерацию в ближайшем внешнем цикле **while**, **do-while** или **for**. Управление передается проверочному выражению циклов **while** и **do-while** или выражению управления циклом в цикле **for**. **Continue** принимает необязательный числовой аргумент, который указывает на скольких уровнях вложенных циклов будет пропущена оставшаяся часть итерации. Значением по умолчанию является 1, при которой пропускается оставшаяся часть текущего цикла.

В примере показана работа оператора **continue** – выводятся элементы массива, стоящие на нечетных местах – 6, 10, 15 (нумерация в массивах PHP начинается с 0).

```
<?php
     $arr = array(5,6,8,10,12,15);
     foreach ($arr as $key => $value) {
          if (!($key % 2)) { // пропуск чётных чисел
               continue;
          }
     print_r($value." ");
     }
?>
```

1.4. Обработка данных форм

Мы уже касались вопроса о том, как языки программирования, в частности PHP, способствовали развитию Интернета и превращению его из статичного, передающего пользователю только ту информацию, которую содержат HTML-страницы, в Интернет динамический, где каждый пользователь, вне зависимости от навыков программирования, может создавать собственный контент: формировать заявку, писать и получать электронные письма, загружать в Интернет файлы и прочее. Передаче информации от пользователя к информационной системе предназначены HTML-формы. Все пользователи Интернета с ними сталкивались – это текстовые поля, поля для заполнения пароля, прикрепление файла, радиокнопки и многое другое.

Рассмотрим форму, показанную на рисунке 7.

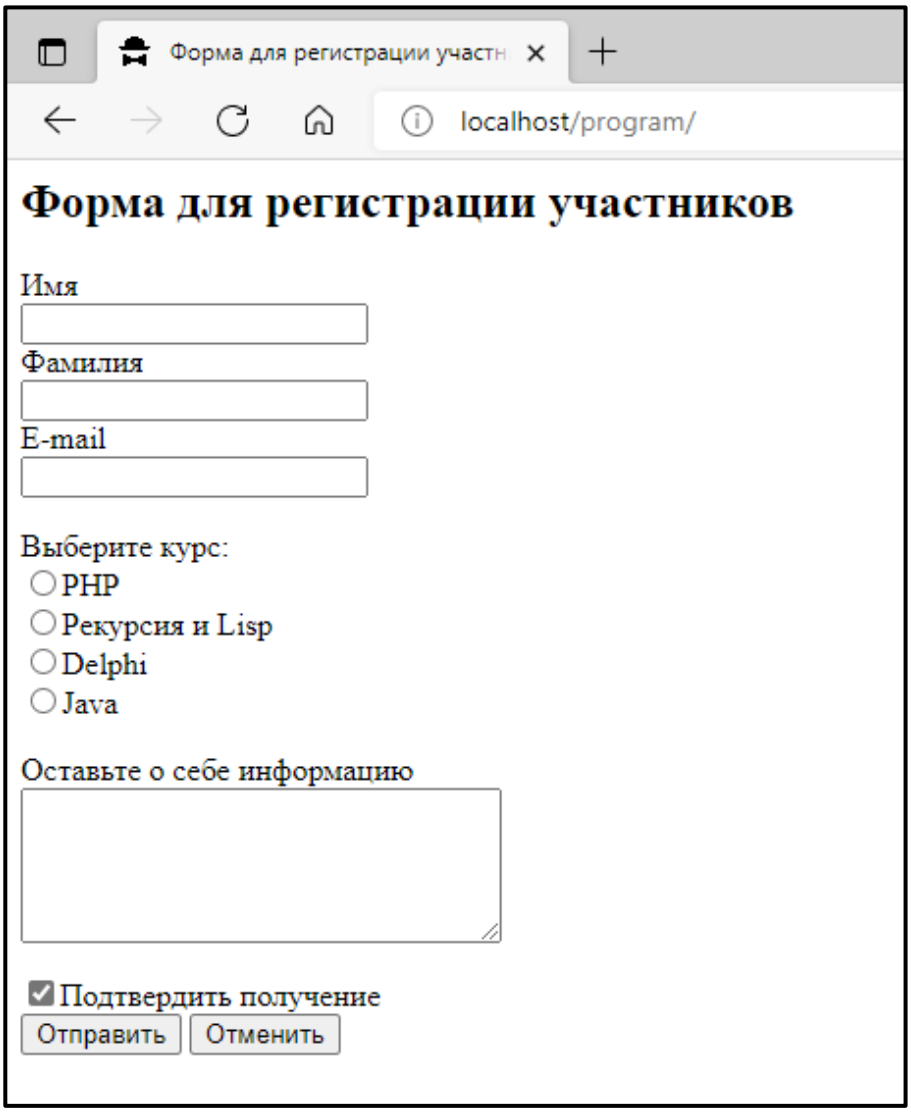


Рис. 7. Пример формы регистрации участников

Через данную форму на удаленный сервер можно передать какие-то данные от пользователя, к примеру необходимые для регистрации участника. Данные с формы отправляются после нажатия на кнопку «Отправить». Если же нажать на кнопку «Отменить», то все данные полей очистятся.

Рассмотрим **общее описание HTML-формы**.

```
<form
     name="имя формы"
     action="путь к программе-обработчику формы"
     method="метод передачи данных">
     Элементы формы или тело формы
</form>
```

Здесь **name** – имя формы. Часто бывает на одной странице размещено несколько форм (например, форма поиска, регистрации участников, форма обратной связи и т.д.). Поэтому каждая форма должна иметь уникальное имя, чтобы скрипты могли получать данные к ее элементам.

Action – атрибут, указывающий URL к программе, которая будет обрабатывать данные с формы (например, записывать в базу данных, отправлять почтовое сообщение, что-то подсчитывать и др.).

Method – метод передачи данных с формы. Используется один из двух методов: или GET или POST. Подробнее о методах передачи информации с форм и их отличиях мы поговорим в конце параграфа.

Можно использовать упрощенную запись формы:

```
<form>
Элементы формы или тело формы
</form>
```

В этом случае программа-обработчик будет являться тем же файлом, который содержит эту форму, метод передачи данных по умолчанию GET.

В тело формы добавляются различные поля. Большая часть из представленных далее полей содержит атрибут **name** (уникальное имя), **type** (тип поля), **value** (значение поля по умолчанию), **title** (всплывающая подсказка). Индивидуальные атрибуты для каждого поля подписаны, подробнее о полях можно прочитать на ресурсе http://htmlbook.ru.

Поле для текстового ввода (рисунок 8).

```
<input
     type="text"
     name="имя элемента"
     value="значение элемента"
     title="всплывающая подсказка"
     size="длина"
     maxlength="максимальное количество элементов">
```

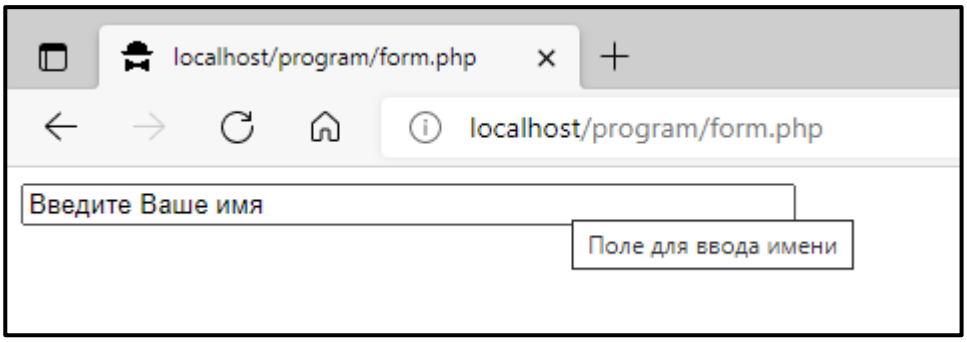


Рис. 8. Пример текстового поля

Поле для ввода пароля – основное отличие от текстового поля в том, что текст в поле заменяется на точки или звездочки.

```
<input
     type="password"
     name="имя элемента"
     value="пароль"
     title="всплывающая подсказка"
     size="длина"
     maxlength="максимальное количество символов">
```

Кнопка используется для отправки данных с форму скрипту обработчику (рисунок 9).

```
<input
          type="submit"
          name="имя элемента"
          value="текст кнопки"
          title="всплывающая подсказка">
```

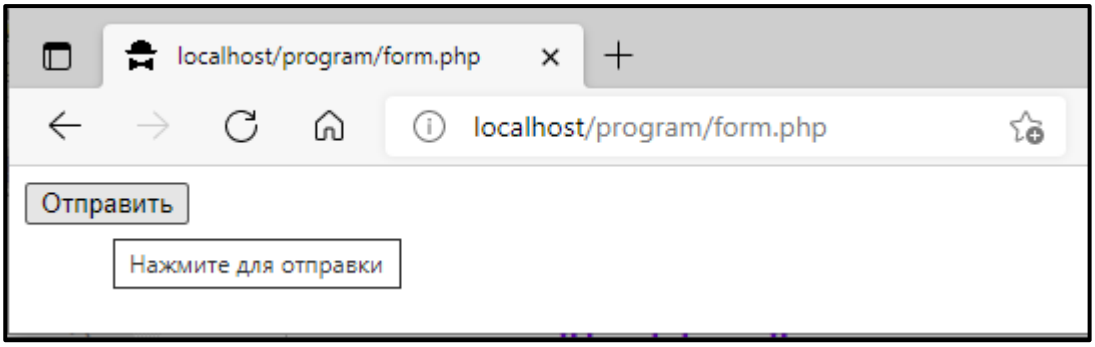


Рис. 9. Пример кнопки

Переключатель или Radiobutton (рисунок 10).

```
<input
     type="radio"
     name="имя элемента"
     value="значение"
     [checked]>
     Текст для переключателя
```

Данное поле содержит необязательный атрибут checked, который может сделать данное поле нажатым по умолчанию. Если используется группа радиопереключателей, то нужно им всем давать одинаковое имя (атрибут name), чтобы в группе переключателей пользователь мог выбрать только один.

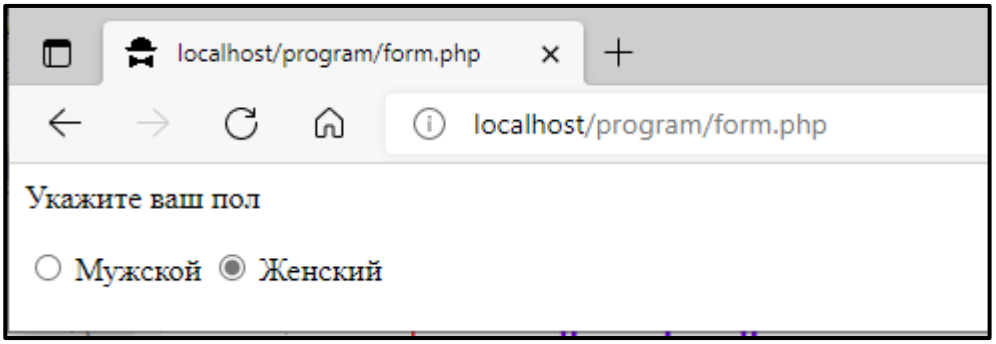


Рис. 10. Пример радио переключателей

Флаг (checkbox) может находиться в одном из двух состояний: отмечен или не отмечен. В отличие от радиокнопки, где из группы может быть выбран только один вариант, в группе чек-боксов может быть выбрано сколько угодно вариантов или вообще ни одного (рисунок 11).

```
<input
     type="checkbox"
     name="имя элемента"
     value="значение"
     [checked]>
     Текст флага
```

Данное поле содержит необязательный атрибут checked, который может сделать данное поле нажатым по умолчанию. Если используется группа флажков, то нужно им всем давать одинаковое имя (атрибут name), чтобы в скрипт-обработчик все выбранные переключатели передавались как элементы массива.

Предположим, что мы хотим предоставить пользователю возможность выбрать курс обучения (рисунок 11).

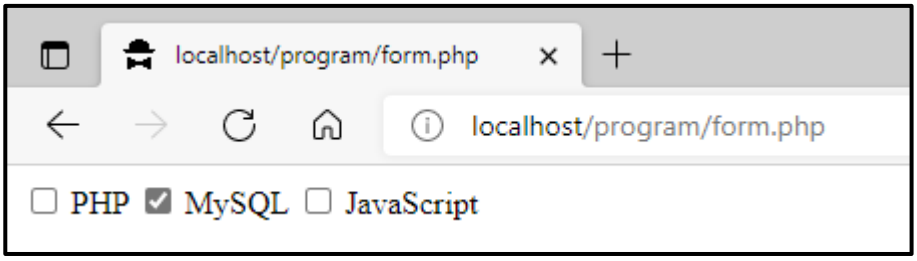


Рис. 11. Пример группы флажков (чек-боксов)

Группа чек-боксов в этом случае выглядит следующим образом:

```
<input
     type="checkbox"
     name="courses[]"
     value="1">
     PHP
<input
     type="checkbox"
     name="courses[]"
     value="2"
     checked>
     MySQL
<input
     type="checkbox"
     name="courses[]"
     value="3">
     JavaScript
```

Например, атрибут name="courses[]" соберет все данные с флагов в массив courses[]. Обратите внимание на наличие квадратных скобок [и]. Без этих скобок переменная courses не будет хранить значения в виде массива и передаст только последний выбранный элемент.

Список элементов позволяет выбрать один или несколько элементов из группы элементов списка (рисунок 12). Общая форма записи элементов списка следующая:

```
<select
     size="количество видимых элементов"
     name="имя списка"
     [multiple]>
     <option value="значение1" [selected]>Отображаемая
строка 1</option>
     …
     <option value="значениеN" [selected]>Отображаемая
строка N</option>
</select>
```

Поле select содержит необязательный атрибут multiple, который позволяет выбрать несколько элементов. Для передачи скрипту-обработчику нескольких значений списка нужно использовать атрибут name с квадратными скобками [и], как это было в случае с чек-боксами.

Каждый элемент списка обрамляется тегами <option>…</option>, у которых также может присутствовать необязательный атрибут selected. В этом случае элемент списка считается выбранным по умолчанию. Если таких элементов несколько, то выбранным в списке отображается последний из них. Необязательный атрибут size со-

держит количество видимых в списке элементов. Это удобно, когда список большой, пользователю удобнее ориентироваться, что предшествует данному элементу, какой следующий и т.д.

Далее на рисунке показаны различные виды списков (выпадающий с выбором одного элемента, открывающий части элементов с выбором только одного, открытый с выбором нескольких элементов).

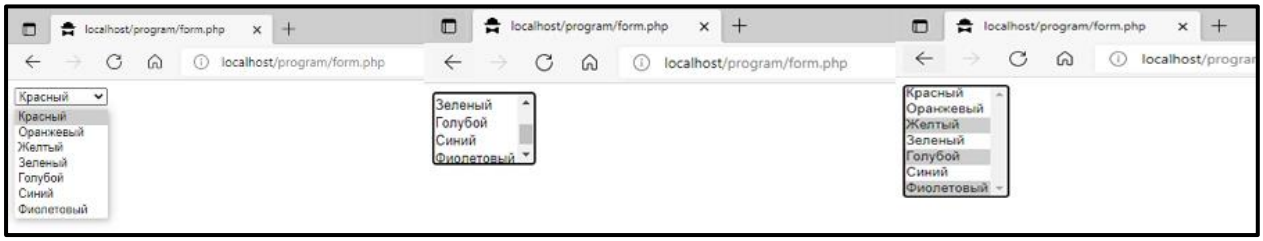


Рис. 12. Пример различных видов списков

Поле для ввода многострочного текста (TextArea) представляет собой элемент формы для создания области, в которую можно вводить несколько строк текста. В отличие от тега **input** в текстовом поле допустимо делать переносы строк, они сохраняются при отправке данных на сервер (рисунок 13).

```
<textarea
     name="имя элемента"
     rows="высота"
     cols="длина">ТЕКСТ ЭЛЕМЕНТА
</textarea>
```

Данный элемент содержит атрибуты rows и cols, которые определяют высоту и ширину поля соответственно в символах.

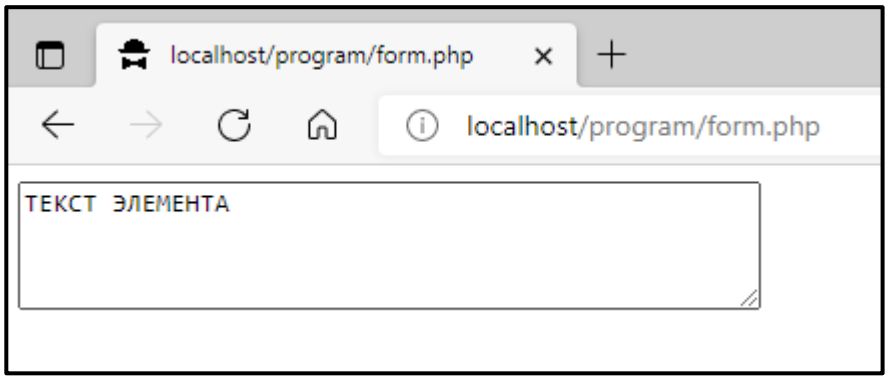


Рис. 13. Пример многострочного текстового поля

Поле для добавления файла предназначено для того, чтобы можно было загружать на сервер один или несколько файлов. Для работы с файлами заголовок формы должен быть сопровожден специальным атрибутом enctype="multipart/form-data" и данные с формы должны передаваться только с помощью метода POST (рисунок 14).

Общий вид поля для добавления файла следующий:

```
<input type="file"
     name="имя поля"
     accept="типы файлов"
     size="ширина поля"
     [multiple]>
```

Как видно из кода, поле содержит необязательный атрибут multiple, который позволяет через одно поле передать сразу несколько файлов. В этом случае атрибут name данного поля должен содержать название с квадратными скобками [и], как это было в случае с чек-боксами и списками.

Также в специальном атрибуте accept можно указать типы файлов, которые можно передавать через данное поле. Например, для передачи только файлов изображений можно использовать следующую запись: `accept="image/*"`.

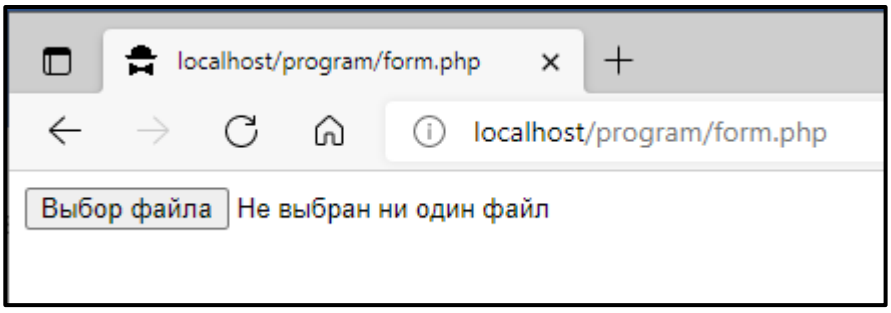


Рис. 14. Пример поля для прикрепления файла

Скрытое поле – предназначено для передачи через форму каких-то данных, параметров, которые нужны разработчику. При этом пользователи этих полей не видят, но в HTML-коде страницы их можно увидеть. Синтаксис данного поля:

```
<input type="hidden"
     name="имя поля"
     value="значение поля">
```

Особенность работы по протоколу HTTP, который является основным при передаче информации от сервера к клиенту, является то, что он не поддерживает состояний. Каждый запрос, который клиент (браузер) отправляет к серверу, является неза-

висимым от других запросов, при этом сервер не запоминает никаких параметров (решение данной проблемы все же будет озвучено в этом пособии).

Все формы, которые были приведены выше, это часть HTML-кода страницы, который связан лишь с отображением форм. А вот передать данные скрипту по назначению HTML не может – это прерогатива языков программирования, в нашем случае PHP.

Данные в виде переменных и их значений передаются от формы к программе с помощью методов. Различают два основных метода: GET и POST.

Метод GET предназначен для открытой передачи данных, то есть любой человек может всегда увидеть, какие данные передаются через строку адреса браузера. Данные методом GET передаются через URL (universal resource locator – адрес ресурса). Рассмотрим пример такого адреса:

```
http://site.ru/index.php?var1=value1&var2=value2
```

Здесь, начиная со знака «?», идет передача значений методом GET в виде пар «имя_переменной=значение». Каждая пара отделяется друг от друга символом амперсанда «&». В данном примере сайту http://site.ru, скрипту index.php передаются две переменные: var1 со значением value1 и var2 со значением value2.

У метода GET есть свои особенности:

- метод GET ограничен отправкой только 1024 символов (то есть передать длинное сообщение или файл через данный метод не получится);
- нельзя использовать метод GET для передачи паролей или любой другой конфиденциальной информации (представьте, если вы ввели в поле пароль, нажали на кнопку «Отправить», и ваш пароль отразился в строке адреса браузера, видимый всем, кто стоит рядом с вами);
- метод GET применим тогда, когда данные нужно запомнить и по возможности передавать по ссылкам. Например, сложный фильтр для получения информации на сайте, поисковый запрос, параметры для доступа к определенным материалам на сайте и др.

Скрипт-обработчик данных с формы (указанные через атрибут action формы) получает данные в специальный суперглобальный массив $_GET. Ключами (индексами) данного массива являются имена переменных. Из приведенного выше примера, скрипт index.php, который находится на домене http://site.ru, получает две переменные следующим образом:

```
$_GET["var1"]   (содержит значение value1)
$_GET["var2"]   (содержит значение value2)
```

Существует также суперглобальный массив $_REQUEST, с помощью которого можно получать все переменные, переданные и методом GET и методом POST.

Рассмотрим еще один пример. Пусть есть файл с формой **form.html** и обработчиком этой формы является файл **handler.php.**

Form.html

```
<form action="handler.php" method="get">
     <input type="text" name="fio" value="">
     <input type="text" name="email" val-
ue="example@mail.ru">
     <input type="submit">
</form>
```

handler.php

```
<?
     $a=$_GET["fio"];
     Echo "Ваше имя $a";
     $b=$_REQUEST["email"];
     Echo "Ваш email $b";
?>
```

Из примера видно, что с формы передаются данные двух текстовых полей fio и email. Скрипт-обработчик получает их либо через массив $_GET, либо через массив $_REQUEST.

Многие разработчики применяют метод GET для передачи данных с форм крайне редко – метод хорошо применяется при передаче данных по гиперссылке, а вот при передаче данных с формы, где требуется написать длинный текст или заполнить множество полей, лучше использовать метод POST.

Особенности метода POST:

- не имеет ограничений по объему отправляемых данных;
- данные, отправленные методом POST, проходят через HTTP-заголовок, поэтому их безопасность зависит от протокола HTTP. Если использовать Secure HTTP (HTTPS), то можно обеспечить защиту передаваемой информации;
- запрос, который был сформирован методом POST в некоторый момент времени, нельзя запомнить и передать в виде ссылки.

Таким образом, можно сказать, что и метод GET и метод POST не являются абсолютно безопасными. Более того, несмотря на то, что метод POST не передает данные в строке адреса, увидеть передаваемые данные через современные браузеры все же можно. Поэтому разработчику при получении данных с форм нужно их филь-

тровать – проверять тип переменной или преобразовывать к нужному типу, убирать из строковых значений посторонние символы и т.д.

Данные методом POST можно получить с помощью ассоциативного массива $_POST, ключами этого массива будут названия передаваемых полей (как и случае с методом GET). Также, с помощью массива $_REQUEST можно получить значения переменных, переданных методом POST.

Рассмотрим пример обработки формы, показанной на рисунке 7 (стр. 27).

index.php

```
<html>
     <head><title>Форма для регистрации участни-
ков</title></head>
     <body>
          <h2>Форма для регистрации участников</h2>
          <form action="1.php" method="post">
          Имя <br><input type="text" name="first_name" ><br>
          Фамилия <br><input type="text" name="last_name"><br>
          E-mail <br><input type="text" name="email"><br>
          <p>
          Выберите курс:<br>
          <input type="radio" name="kurs" value="PHP">PHP<br>
          <input type="radio" name="kurs"
value="Lisp">Рекурсия и Lisp<br>
          <input type="radio" name="kurs" val-
ue="Delphi">Delphi<br>
          <input type="radio" name="kurs" val-
ue="Java">Java<br>
          <P>Оставьте о себе информацию <BR>
          <textarea name="comment" cols="32"
rows="5"></textarea>
          <P><input name="confirm" type="checkbox"
checked>Подтвердить получение <br>
          <input type="submit" name="submit" val-
ue="Отправить">
          <input type="reset" value="Отменить">
          </form>
     </body>
</html>
```

Скриптом-обработчиком формы регистрации участников является файл **1.php**

```
<?php
$submit = $_POST["submit"];
if (isset($submit)){
     $str = "Здравствуйте,
          ".$_REQUEST["first_name"]. "
          ".$_REQUEST["last_name"]."! <br>";
```

```
        $str .="Вы выбрали для изучения курс по
             ".$_REQUEST["kurs"];
        echo $str;
}
?>
```

Здесь важно сделать простую проверку, нажал ли пользователь на кнопку: если существует переменная $submit, полученная методом POST (непустое значение кнопки «Отправить»), то можно получить остальные данные с формы (в коде скрипта их получаем с помощью массива $_REQUEST) и произвести с ними какие-то действия. Результат работы скрипта 1.php может выглядеть в браузере следующим образом (рисунок 15).

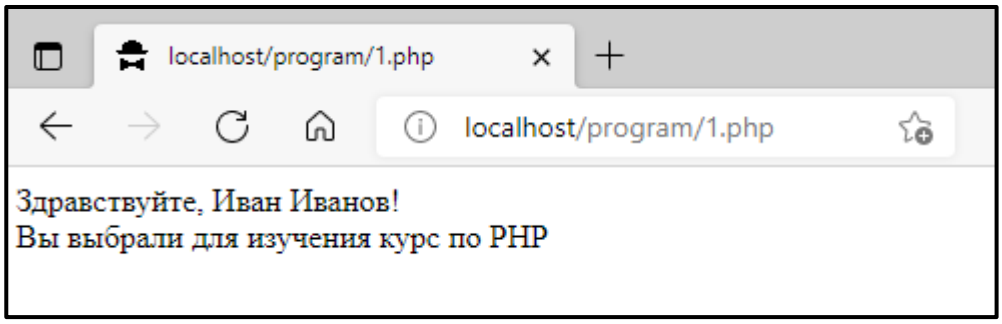


Рис. 15. Пример результата обработки данных с формы регистрации участника

Таким образом, язык разметки гипертекста HTML и язык программирования PHP дополняют друг друга при передаче данных от клиента (браузера) на сервер. HTML предоставляет теги для формирования форм на веб-странице, а PHP обрабатывает данные, передаваемые с форм, с помощью методов GET и POST. При передаче данных на сервере их можно получить с помощью суперглобальных массивов $_GET, $_POST или все сразу с помощью $_REQUEST.

1.5. Сессии и cookie в PHP

Как уже было сказано ранее, протокол HTTP не может передать состояние и параметры. Поэтому каждый отдельный HTTP-запрос независим от других, сервер обрабатывает каждый такой запрос отдельно. Частично, эту проблему решает передача методами GET и POST. Однако, при разработке веб-проектов часто возникает необходимость передать какие-то данные между файлами (скриптами). Типичная ситуация – представьте себе сайт, на котором есть форма авторизации пользователя. Естественно, вводя свои данные одни раз, пользователь не ожидает, что при переходе на другие страницы данного ресурса ему снова придется вводить логин и пароль. Однако все вышеописанные средства для решения этой задачи не подошли бы.

Опишем данную проблему более формально. Пусть имеется файл first.php, в котором мы задаем значение какой-либо переменной.

Файл **first.php**

```
<?php
  $a = "Меня задали на first.php";
?>
<html>
     <body>
          <?php
          echo $a;
          ?>
     </body>
</html>
```

И предположим, что эту переменную нам необходимо передать в файл second.php. Но файл second.php ничего не знает о значении переменной $a.

Файл **second.php**

```
<html>
     <body>
<?php
  echo $a;
?>
     </body>
</html>
```

Именно из-за отсутствия поддержки передачи информации между скриптами, файл **second.php** выведет пустое значение переменной **$a**.

Решением данной проблемы является механизм **сессий** или **сеансов**. Условно говоря, сессией является сеанс связи конкретного лица, сидящего за конкретным компьютером (или цифровым устройством), который с помощью одной и той же программы-браузера посещает ресурсы удаленного сервера. Если пользователь закрыл браузер, то сеанс его связи будет прерван. Если тот же самый пользователь, с того же самого устройства заходит на те же ресурсы, но с другого браузера, то это будет новый сеанс (сессия). Формально в начале сеанса связи сервер определяет небольшую область под хранение данных конкретного сеанса. Способ хранения этих данных (в виде файлов или в виде записей в базе данных) определяется в настройках сервера.

Любой скрипт, который будет использовать переменные (данные) из сессий, должен содержать следующую строчку:

```
session_start();
```

Эта команда говорит серверу, что данная страница нуждается во всех переменных, которые связаны с данным пользователем (браузером). Сервер берёт эти переменные (из файла либо из БД) и делает их доступными.

Очень важно открыть сессию до того, как какие-либо данные будут посылаться пользователю; на практике это значит, что функцию **session_start()** желательно вызывать в самом начале страницы (до тега <html>), например так:

```
<?php
  session_start();
?>
<html>
 <head>
 </head>
```

После того как применен вызов функции **session_start()** и установлен сеанс, можно применить простой подход к чтению/записи сессионных переменных через суперглобальный массив **$_SESSION**. Еще раз подчеркнем, что сессионные значения хранятся на стороне сервера и хранятся до тех пор, пока сеанс связи не прервется (пользователь не закроет браузер или длительность сессии можно запрограммировать). При присвоении какого-либо значения любому полю массива **$_SESSION**, переменная с таким же именем автоматически регистрируется как переменная сессии. Этот массив доступен на всех страницах, использующих сессию.

Рассмотрим, как изменится код скриптов при решении задачи передачи значения переменной **$a** между скриптами **first.php** и **second.php**.

Файл **first.php**

```
<?php
     // открываем сессию
     session_start();
     // задаём значение переменной
     $a = "Меня задали на first.php";
     // регистрируем переменную с открытой сессией
     $_SESSION["a"] = 'Меня задали на first.php'
?>
<html>
     <body>
          Произошел старт сессии
          Перейдите по ссылке для просмотра <a
href="second.php">second.php</a>
     </body>
</html>
```

Файл **second.php**

```
<?php
  // открываем сессию
  session_start();
?>
<html>
     <body>
          <?php
          echo $_SESSION["a"];
          ?>
     </body>
</html>
```

В этом случае можно убедиться, что second.php выведет значение «Меня задали на first.php». При этом сессионный механизм работает, т.к. и в файле first.php и в файле second.php произошло подключение к сеансу через session_start().

Другие функции для работы с сессиями:

- unset($_SESSION['a']) – удаление значение переменной **a** из сессии;
- session_destroy() – сессия уничтожается (например, если пользователь покинул систему, нажав кнопку «Выход»);
- session_set_cookie_params(int lifetime [, string path [, string domain]]) – с помощью этой функции можно установить, как долго будет «жить» сессия, задав unix_timestamp, определяющий время «смерти» сессии. По умолчанию сессия «живёт» до тех пор, пока клиент не закроет окно браузера.

Рассмотрим классический пример применения сессии при авторизации пользователя. Простая авторизация предназначена для выяснения доступа пользователя к некоторым защищенным (засекреченным) страницам. Чаще всего для проверки пользователя используется пара «логин – пароль». Пусть есть пользователь с логином `maria` и паролем `123456` (не самая надежная пара, но для примера подойдет). На блок-схеме показано (рисунок 16), как работает алгоритм.

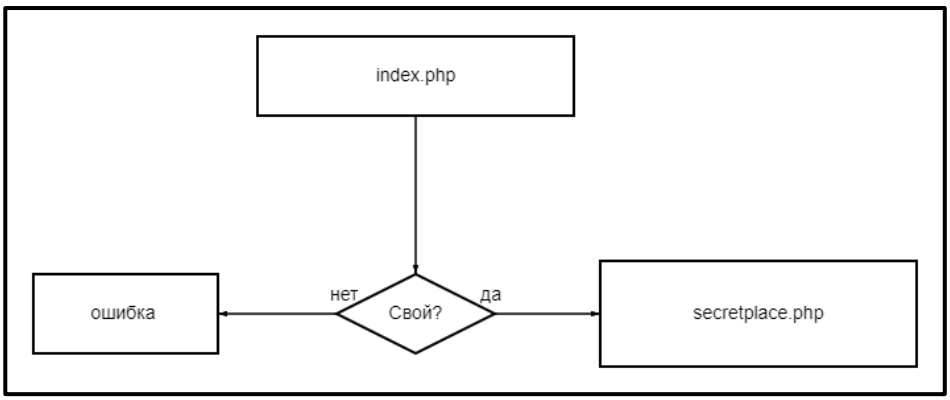


Рис. 16. Блок-схема авторизации пользователя

Файл **index.php**

```
<html>
     <head>
          <title>Введите пароль</title>
     </head>
     <body>
          <form action="authorize.php" method="post">
               Логин:<input type="text" name="username"><br>
               Пароль:<input type="password"
name="password"><br>
               <input type="submit" name="Submit">
          </form>
     </body>
</html>
```

Файл **authorize.php**

```
<?php
  // открываем сессию
  session_start();
  //получаем данные с формы
  $Submit      = $_POST["Submit"];
  $username    = $_POST["username"];
  $password    = $_POST["password"];
// данные были отправлены формой?
  if($Submit){
    // проверяем данные на правильность...
     if(($username=="maria")&&($password=="123456")){
          // запоминаем имя пользователя
          $_SESSION["logged_user"] = $username;
          // и переправляем его на <секретную> страницу...
          header("Location: secretplace.php");
          exit;
    }
  }
  // если что-то было не так, то пользователь получит сообще-
ние об ошибке.
?>
<html><body>
Вы ввели неверный пароль!
</body></html>
```

Рассмотрим код файла **secretplace.php**. Просто зайти на эту страницу нельзя: если имя пользователя не зарегистрировано, то перенаправляем его на страницу index.php для ввода логина и пароля.

Файл **secretplace.php**

```
<?php
  // открываем сессию
  session_start();
  $logged_user = $_SESSION["logged_user"];
  if(!isset($logged_user)){
    header("Location: index.php");
    exit;
  }
?>
<html>
 <body>
 Здравствуйте, <?php echo $logged_user; ?>, вы на секретной
странице<br>
 <a href="logout.php">Выйти</a>
 </body>
</html>
```

Также предусмотрим файл **logout.php**, который позволит выйти из сеанса текущего пользователя (сбросить авторизацию). При этом для чистоты эксперимента после уничтожения сессии перенаправим пользователя снова на секретную страницу. Если пользователь не авторизован, то уже скрипт **secretplace.php** перенаправит пользователя на **index.php**.

Файл **logout.php**

```
<?php
  session_start();
  //уничтожаем сессию
  session_destroy();
  //перенаправляем на секретную страницу,
  //если сессия сброшена, то произойдет перенаправление на
  //index.php
  header("location: secretplace.php");
 ?>
```

Увидеть файлы с данными сессии можно на локальном сервере в директории WebServers/tmp. Все файлы в этой папке имеют специальные идентификаторы, но если их отсортировать по дате, то можно легко найти последние изменённые файлы (рисунок 17).

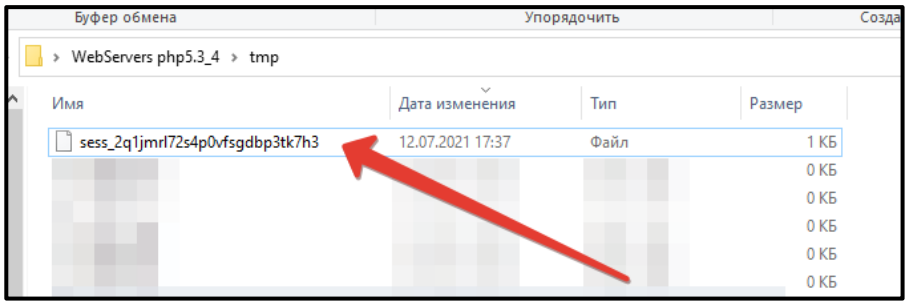


Рис. 17. Список файлов сессий

Содержимое этого файла можно легко посмотреть с помощью текстового редактора (рисунок 18).

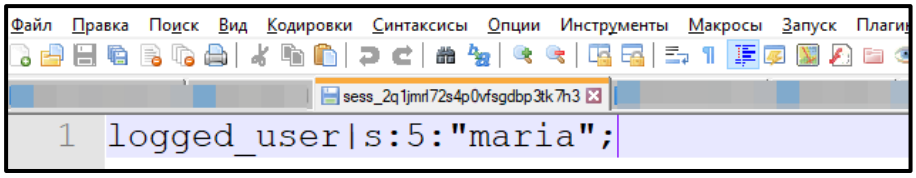


Рис. 18. Содержимое сессионного файла

Для отслеживания сессий часто применяются специальные **cookie-файлы**. Они хранятся на компьютере пользователя в виде небольшой текстовой информации. В каждом браузере своя директория для хранения cookie-файлов. В отличие от файлов сессий, cookie могут храниться дольше закрытия браузера. Более того, можно определять время хранения файлов cookie программно. Cookie-файл хранит данные в виде пар «имя – значение».

Cookie применяются для контроля за некоторыми фрагментами информации, к которой обращается пользователь при переходе от страницы к странице. При этом предполагается, что эта информация хранится и после того, как пользователь прервал сеанс связи с сайтом. Примером использования cookie может служить алгоритм голосования – с помощью cookie можно запретить пользователю принимать участие в голосовании более одного раза. Чуть позже мы разберем подробнее этот пример.

Для установки cookie-файлов используется функция setcookie().

```
Setcookie (имя_cookie, значение_cookie, вре-
мя_жизни_cookie_в_секундах);
```

Как и любой другой заголовок cookie должны передаваться **до того,** как будут выведены какие-либо другие данные скрипта (это ограничение протокола). Это значит, что в скрипте вызовы этой функции должны располагаться **до остального вывода**, включая вывод тегов <html> и <head>, а также пустые строки и пробельные символы.

Пример

```
setcookie ('my_cookie', 'Понедельник', time()+60*60*24*3);
```

Функция **time()** – возвращает количество секунд, прошедших с 1 января 1970 года. Обратите внимание, что использование записи 60*60*24*3 в качестве добавления секунд жизни cookie гораздо информативнее, чем вычислить это значение и записать 259 200. Так сразу понятно, что время жизни cookie трое суток.

Считывание переменных cookie может происходить с помощью суперглобального массива $_COOKIE (по аналогии с суперглобальным массивом $_SESSION).

Вернемся к примеру с голосованием. Одной из наиболее частных проблем при сборе сведений через формы голосования является то, что один и тот же пользователь может многократно проголосовать. В некоторых случаях это существенно влияет на результаты голосования, поэтому разработчику нужен механизм запрета голосования для пользователя, который уже ранее голосовал. Механизм сессий здесь не подходит, так как сессия уничтожается, как только пользователь закрыл браузер, и с точки зрения сервера – это будет новый респондент. А вот использование cookie здесь как раз пригодится. После того как пользователь первый раз проголосовал, в его cookie сохраняется небольшая информация. При попытке в следующий раз проголосовать скрипт сначала посмотрит, не сохранено ли что-то насчет голосования в cookie, если нет, то позволит проголосовать, если есть, то не позволит. Конечно, cookie-файл привязан к браузеру, и ничто не мешает этому же пользователю проголосовать через другой браузер. Но пользователь о таком скорее всего не знает, и в большинстве своем метод сработает.

Фрагмент алгоритма голосования

```
if (!(isset($_COOKIE['golos']))) {
  setcookie('golos', 'yes', time()+60*5);
  }
 else { echo "Вы уже голосовали";  }
```

В данном случае пользователь не может проголосовать в течение пяти минут.

Таким образом, сессии и cookie позволяют отслеживать и запоминать действия пользователей при взаимодействии с сайтом. Если нужно менять информацию пользователю в зависимости от его выбора или ранее сделанных действий (посещенных страниц, авторизации, голосования, отбора продуктов в корзину, заполнении форм обратной связи и др.), то можно применять сессии и cookie. Надо понимать, что по-

мимо удобства для пользователя, эти механизмы должны быть надежно защищены от несанкционированного доступа посторонних лиц.

1.6. Работа с функциями

В программировании, как и в математике, **функция** есть отображение множества ее аргументов на множество ее значений. То есть функция для каждого набора значений аргумента **возвращает** какие-то значения, являющиеся **результатом** ее работы.

Как и в любом другом языке программирования, в PHP различают функции, определяемые пользователем (в данном случае программистом), и функции, которые заранее определены в языке программирования. Рассмотрим их по очереди.

Функции, определяемые пользователем, это законченные части программ, которые решают определенную задачу. Пусть, к примеру, необходимо подсчитать факториал какого-либо числа. Факториал числа есть произведение всех чисел от единицы до этого числа. Если брать в качестве числа небольшое, однозначное число, например 4, то задача будет очень простой: 4! = 1*2*3*4 = 24. Но если взять число побольше, например 10, то выписывать все умножения уже несколько утомительно. Функции как раз и предназначены для решения типовой задачи, при этом параметры функции можно менять. Пример алгоритма вычисления факториала числа на PHP приведен ниже.

```
<?php
function fact($n){
  if ($n==0) return 1;
  else return $fact = $n * fact($n-1);
}
echo fact(3);
   // можно было бы написать echo (3*2);
   // но если число большое,
echo fact(50);
   // то удобнее пользоваться функцией,
   // чем писать echo (50*49*48*...*3*2);
?>
```

Описание функции начинается с служебного слова function. В PHP не важно, где именно описывается функция – до или после вызова. После слова function идет название функции, которое должно быть уникальным и содержать символы латин-

ского алфавита. Название не должно начинаться с цифр, служебных символов, пробелов. В имени функции допускается использование нижнего подчеркивания. Если функция содержит входные параметры, то они должны быть определены сразу после имени функции. Внутри функции описываются действия. Если функция должна возвращать какое-то значение, то в теле функции должен быть оператор return, выполняющий эту работу. Пример общего описания функции представлен ниже.

```
function Имя_функции (параметр1, параметр2, ..., параметрN){
    Блок_действий
    return "значение возвращаемое функцией";
}
```

Для того чтобы функция выполнила поставленную ей задачу, ее необходимо вызвать. Вызов функции может осуществляться как до, так и после ее описания. Единственное ограничение в этом случае – описание функций внутри условных конструкций. В случае, когда функция определяется в зависимости от какого-либо условия, например как это показано в двух приведённых ниже примерах, обработка описания функции должна **предшествовать** её вызову (подробнее можно изучить на ресурсе [18]). Вызов функции осуществляется указанием ее имени и значений параметров:

```
<?php
Имя_функции("значение_параметра1","значение_параметра2",...);
?>
```

Аргументы функции бывают двух видов: аргументы-значения и аргументы-переменные.

Аргументы-значения не меняют своего значения в процессе выполнения функции. В этом случае при вызове функции на их месте могут находиться конкретные значения, которые будут использованы в функции. Когда аргумент передается в функцию по значению, изменение значения аргумента внутри функции не влияет на его значение вне функции.

Чтобы позволить функции изменять ее аргументы, их нужно передавать по ссылке (аргументы-переменные). Для этого в определении функции перед именем аргумента следует написать знак амперсанд «&».

```
<?php
//напишем функцию, которая бы добавляла к строке слово se-
lected
function addOption(&$option){
     $option .= "selected";
}
//выводит элемент формы список
$str = "<select>";
echo $str;
// пусть имеется такая строка
$option = "<option ";
echo $option .">Красный</option>";
// выведет "красный", но он не будет выделен
addOption($option);
// вызовем функцию
echo $option .">Синий</option>";
//это выведет в списке выделенный "синий"
$str = "</select>";
echo $str;
?>
```

При работе с функциями в PHP есть замечательная особенность – задавать аргументы функции, используемые по умолчанию. В тех случаях, когда в других языках программирования необходимо дополнительно делать проверки условий существования переданных значений, в PHP можно задавать значения параметров непосредственно в заголовке функции. Само значение по умолчанию должно быть константным выражением, а не переменной и не представителем класса или вызовом другой функции.

Ниже представлена функция, создающая информационное сообщение, подпись к которому меняется в зависимости от значения переданного ей параметра. Если значение параметра не задано, то используется подпись "ЮУрГГПУ".

```
<?php
function message($sign="ЮУрГГПУ."){
// здесь параметр sign имеет по умолчанию значение "ЮУрГГПУ"
echo "Вы приглашены на семинар по веб-технологиям.<br>";
echo "$sign<br>";
}
```

```
message();
//вызываем функцию без параметра.
//В этом случае подпись – это ЮУрГГПУ.
message("С уважением, Иванов В.П.");
// В этом случае подпись "С уважением, Иванов В.П."
?>
```

Если у функции несколько параметров, то те аргументы, для которых задаются значения по умолчанию, должны быть записаны после всех остальных аргументов в определении функции. В противном случае появится ошибка, если эти аргументы будут опущены при вызове функции. Код функции А

```
<?php
function addArticle($title, $description, $author="Иванов Иван"){
  echo "Заносим в каталог статью: $title,";
  echo "автор $author";
  echo "<br>Краткое описание: ";
  echo "$description <hr>";
}
addArticle("Информатика и мы", "Это статья про информатику...", "Петров Петр");
addArticle("Кто такие продакт-менеджеры", "Это статья про менеджеров проектов...");
?>
```

Пример **неправильной** работы с параметрами функции (значение параметра по умолчанию находится перед остальными параметрами):

```
<?php
function addArticle($author="Иванов Иван", $title, $description){
// ...действия как в предыдущем примере
}
addArticle("Кто такие продакт-менеджеры", "Это статья про менеджеров проектов...");
?>
```

Рассмотрим работу с областью видимости переменных, используемых как в основной программе, так и в области функций. К таким областям видимости относят глобальные и статичные переменные.

Чтобы использовать внутри функции переменные, заданные вне ее, эти переменные нужно объявить, как **глобальные**. Для этого в теле функции следует перечислить их имена после ключевого слова **global.**

Пример использования глобальных переменных

```
<?
$a=1;
function test_g(){
global $a;
  $a = $a*2;
  echo 'в результате работы функции $a=', $a;
}
echo 'вне функции $a=', $a, ', ';
test_g();
echo "<br>";
echo 'вне функции $a=', $a, ', ';
test_g();
?>
```

В результате работы этого примера будет выведено:

```
вне функции $a=1, в результате работы функции $a=2
вне функции $a=2, в результате работы функции $a=4
```

Статические переменные видны только внутри функции и не теряют своего значения, если выполнение программы выходит за пределы функции. Объявление таких переменных производится с помощью ключевого слова **static.**

Пример работы со статическими переменными

```
<?
function test_s(){
static $a = 1;
// нельзя присваивать выражение или ссылку
  $a = $a*2;
  echo $a;
}
test_s(); // выведет 2
echo $a;  // ничего не выведет, так как $a доступна
```

```
            //только внутри функции
test_s(); // внутри функции $a=2, поэтому
          // результатом работы функции
          // будет число 4
?>
```

Если функция в результате работы должна возвращать один результат, то используют оператор **return**. Отметим, что return может возвращать как одно значение, так и несколько значений в виде массива. Тем не менее считается, что функция возвращает один результат – массив.

Рассмотрим пример возвращения функцией количества дней, прошедших между двумя датами. Причем неважно, какая из дат наступила раньше.

```
<?php
function daysBetween($start_data, $end_data)
{
return
abs(strtotime($start_data)-strtotime($end_data))/(60*60*24);
}
$how_days = daysBetween("14.03.2021", "17.03.2021");
echo "Прошло $how_days дней";
?>
```

Здесь мы сталкиваемся с функцией, которая уже определена в php – **strtotime**. Ее задача – перевести строковое значение даты в количество секунд, прошедших с 1 января 1970 года 00:00:00 UTC (так называемое начало эпохи UNIX). Математическая функция **abs** возвращает модуль числа, в этой задаче она служит для вычисления абсолютной величины между двумя датами.

Рассмотрим также и другие популярные функции, уже определенные в PHP. Все функции разбиты по категориям – для работы с массивами, для работы со строками, для работы с датой/временем, для работы с mysql и другие. Большинство функций в PHP имеет особую приставку (префикс) в названии, с помощью которой разработчику легко определить ее применимость и назначение.

Кратко рассмотрим функции для работы с массивами, строками и ряд других, которые будут использованы на лабораторных занятиях в рамках данного курса. Про все неописанные функции можно узнать из источников [18; 20].

Многие функции для работы с массивами имеют префикс «array_». Самыми популярными функциями являются:

- count($array) – возвращает количество элементов массива.
- in_array($elem, $array) – проверяет, содержится ли элемент $elem в массиве $array.
- array_search($elem, $array) – проверяет, содержится ли элемент $elem в массиве $array, и возвращает индекс найденного элемента.
- array_values($array) – возвращает новый массив, содержащий все значения переданного на вход массива.
- array_keys($array) – возвращает новый массив, содержащий все индексы в качестве значений переданного на вход массива.
- sort($array, [$flag]) – сортирует массив $array в зависимости от необязательного параметра $flag. Если значение $flag задать SORT_REGULAR, то произойдет обычное сравнение элементов; если SORT_NUMERIC, то числовое сравнение элементов; если SORT_STRING, то строковое сравнение элементов.
- rsort($array, [$flag]) – выполняет сортировку массива в обратном порядке.
- asort($array, [$flag]) – простая сортировка с сохранением ассоциаций ключей.
- arsort($array, [$flag]) – обратная сортировка с сохранением ассоциаций ключей.
- ksort($array, [$flag]) – сортировка массива по ключам.
- krsort($array, [$flag]) – обратная сортировка массива по ключам.

Функции для работы со строками часто в своем названии имеют приставку str.

- str_replace($search, $replace, $subject, [$count]) – заменяет все вхождения строки поиска на строку замены. Эта функция возвращает строку или массив, в котором все вхождения $search в $subject заменены на $replace. При этом $search и $replace могут быть как строками, так и массивами строк. Результат функции также может быть строкой или массивом. Необязательный параметр $count может быть использован для возвращения количества произведённых замен.
- substr($string, $offset, [$length]) – возвращает подстроку строки string, которая начинается с номера символа $offset и имеет длину $length символов.

Если $length положительный, возвращаемая строка будет не длиннее $length-символов, начиная с параметра $offset (в зависимости от длины $string). Если $length отрицательный, то будет отброшено указанное этим аргументом число символов с конца строки string (после того как будет вычислена стартовая позиция, если $offset отрицателен). Если при этом позиция начала подстроки, определяемая аргументом $offset, находится в отброшенной части строки или за ней, возвращается false.

- strlen($string) – возвращает длину строки (количество символов в строке).
- explode($separator, $string, [$limit]) – возвращает массив строк, полученных разбиением строки $string с использованием $separator в качестве разделителя. Если аргумент $limit является положительным, возвращаемый массив будет содержать максимум $limit элементов при этом последний элемент будет содержать остаток строки $string. Если параметр $limit отрицателен, то будут возвращены все компоненты, кроме последних $limit. Если $limit равен нулю, то он расценивается как 1.
- implode($separator, $array) – объединяет элементы массива $array в строку с помощью разделителя $separator. В результате возвращает строку.
- strpos($haystack, $needle, [$offset]) – ищет позицию первого вхождения подстроки $needle в строку $haystack. Если $offset указан, то поиск будет начат с указанного количества символов с начала строки. Если задано отрицательное значение, отсчёт позиции начала поиска будет произведён с конца строки.
- addslashes($string) – возвращает строку с обратным слешем перед символами, которые нужно экранировать. Экранируются следующие символы: одинарная кавычка ('); двойная кавычка (''); обратный слеш (\); NUL (байт null).
- trim($string,[$charlist]) – удаляет пробелы из начала и конца строки. Помимо пробелов также может удалять непечатные символы: табуляцию «\t», перевод строки «\n», возврат каретки «\r», NUL-байт «\0», вертикальную табуляцию «\x0B». Можно перечислить удаляемые символы через строку $charlist.
- stripslashes($string) – удаляет экранирующие обратные слэши. (\' преобразуется в ', и т.д.). Двойные бэкслэши (\\) преобразуется в одиночные (\).
- htmlspecialchars($string, []) – преобразует специальные символы в HTML-сущности. '&' (амперсанд) преобразуется в '&'; '"' (двойная кавычка) преобразуется в '"'; ''' (одиночная кавычка) преобразуется в '''

только в режиме ENT_QUOTES; '<' (знак «меньше, чем») преобразуется в '<'; '>' (знак «больше, чем») преобразуется в '>'.

- md5($string) – возвращает MD5-хеш строки в виде 32-символьного шестнадцатеричного числа. Алгоритм необратим, то есть хеш нельзя дешифровать. Функция используется для сравнения файлов, хранения паролей в закодированном виде и других криптографических задач.

Также в PHP есть функции для работы с переменными – проверка значений переменных

- isset($variable) – определяет, была ли установлена переменная $variable значением отличным от null.
- unset($variable) – удаляет переменную $variable.
- intval($variable) – функция преобразования типов, возвращает целое значение переменной $variable.
- floatval($variable) – функция преобразования типов, возвращает число с плавающей точкой по переменной $variable.

Математические функции

- abs($number) – возвращает абсолютное значение $number.
- round($num, [$precision], [$mode]) – возвращает округлённое значение $num с указанной точностью $precision (количество цифр после запятой). $precision может быть отрицательным или нулём (по умолчанию).
- ceil($num) – возвращает ближайшее большее целое от $num.
- mt_rand([$min], [$max]) – генерирует случайное значение методом с помощью генератора простых чисел. $min – необязательный параметр: минимальное значение случайного числа (по умолчанию: 0). $max – необязательный параметр: максимальное значение случайного числа

Функция для отправки почтового сообщения

- mail($to, $subject, $message, [$additional_headers], [$additional_params]) – отправляет электронную почту на адрес $to, с темой $subject и содержанием письма $message. Необязательный параметр $additional_headers – это строка или массив, которые будут вставлены в конец отправляемых заголовков письма. Параметр $additional_params может быть использован для передачи дополнительных флагов в виде аргументов командной строки для программы, сконфигурированной для отправки писем, указанной директивой sendmail_path. Например, можно установить отправителя письма при использовании sendmail с помощью опции -f.

Здесь приведена лишь малая часть функций, которые определены в PHP. В следующем параграфе мы отдельно рассмотрим функции для работы с файловой системой. Функции для работы с базами данных также рассмотрим в параграфе 2.9. Тем не менее даже такая небольшая часть функций может принести большую пользу разработчику, так как они решают наиболее частые задачи при программировании веб-проектов.

1.7. РАБОТА С ФАЙЛОВОЙ СИСТЕМОЙ

При разработке веб-проектов важной задачей является работа с файловой системой. При этом разработчик должен писать алгоритм как для взаимодействия с файловой системой локального компьютера, так и с файловой системой веб-сервера. Все такие операции, как: загрузка файла с локального компьютера на сервер, копирование файла на сервере из одной директории в другую, создание файла, переименование, удаление, редактирование и другие – реализуется с помощью особых функций в PHP. Рассмотрим эти функции:

- copy($source, $dest) – копирует файл $source в файл с именем $dest. Функция возвращает true в случае успешного завершения, или false, в случае возникновения ошибки.
- move_uploaded_file($filename, $destination) – функция проверяет, является ли файл $filename загруженным на сервер (переданным по протоколу HTTP POST). Если файл действительно загружен на сервер, он будет перемещён в место, указанное в аргументе $destination. В отличие от функции copy() эта функция перемещает файл с локального компьютера на сервер.
- rename($oldname, $newname) – пытается переименовать $oldname в $newname, перенося файл между директориями, если необходимо. Если $newname существует, то он будет перезаписан. При переименовании директории с существующим $newname будет выведено предупреждение.
- fopen ($filename, $mode) – открывает файл $filename на операцию в зависимости от значения $mode. $mode – это строка, которая может содержать одно из следующих значений.
  - r – открыть только для чтения, помещает указатель на начало файла.
  - r+ – открыть для чтения и для записи, помещает указатель на начало файла.
  - w – открыть только для записи, помещает указатель на начало файла и очищает все содержимое файла. Если файл не существует, то создается новый.

- w+ – открыть для чтения и для записи, помещает указатель на начало файла и очищает все содержимое файла. Если файл не существует, создается новый.
- a – открыть только для записи, помещает указатель на конец файла. Если файл не существует, создается новый.
- a+ – открыть для чтения и для записи, помещает указатель на конец файла. Если файл не существует, создается новый.

- fclose($filename) – закрывает файл, который ранее был открыт дескриптором fopen(). Возвращает true, в случае успешного завершения, или false, в случае возникновения ошибки.
- feof($filename) – проверка, не конец ли файла. Возвращает true, если указатель файла указывает на EOF или произошла ошибка, иначе возвращает false.
- fgets($handle, [$length]) – читает строку из файлового указателя $handle. $length – необязательное числовое значение, указывающее размер получаемой строки в байтах.
- fread($stream, [$lenght]) – бинарно-безопасное чтение файла. Читает до $length байт из файлового указателя stream и смещает указатель. Чтение останавливается, как только было достигнуто одно из следующих условий: было прочитано $length байт; достигнут EOF (конец файла).
- fwrite($handle, $string, [$length]) – бинарно-безопасная запись в файл. Строку $string функция записывает в файловый поток $handle. Если передан аргумент $length, запись остановится после того, как $length байтов будут записаны или будет достигнут конец строки $string, смотря на то, что произойдёт раньше.
- file($filename) – читает содержимое файла $filename и помещает его в массив. Возвращает файл в виде массива. Каждый элемент массива соответствует строке файла, с символами новой строки включительно. В случае ошибки file() возвращает false.
- file_get_contents($filename) – читает содержимое файла $filename и помещает его в строку.
- unlink($filename) – удаляет файл $filename.

Рассмотрим простой пример работы с файловой системой на PHP – алгоритм открывает файл newname.txt на чтение, затем считывает из этого файла все строки длиной 3 символа и выводит их на экран браузера. После этого происходит безопасное закрытие файла.

```
<?php
if ($fp=fopen("newname.txt","r"))
echo "Работа функции fopen() произведена успешно !<br>";
do {
     $str = fgets ($fp, 3);
     echo $str, " <br>";
}
while (!feof($fp));
if (!fclose($fp)){
     echo "Функция fclose () выполнила ошибку<br>";
}
else {
     echo "Закрытие файла newname.txt осуществлено успеш-
но<br>";
}
?>
```

Рассмотрим еще одну типичную задачу работы с файлами – чтение из файла и получение данных в виде строки.

```
<?php
     //получает содержимое файла в строку
     $filename = "/usr/local/something.txt";
     $handle = fopen($filename, "r");
     $contents = fread($handle, filesize($filename));
     fclose($handle);
?>
```

Помимо чтения необходимо часто использовать операцию запись в файл. При этом в рассмотренном ниже примере запись в файл происходит первых 7 символов.

```
     <?php
     $h = fopen("my_file.html","a");
     $add_text = "Добавим текст в файл.";
     if (fwrite($h,$add_text,7))
          echo "Добавление текста прошло успешно<br>";
     else
          echo "Произошла ошибка при добавлении данных<br>";
          fclose($h);
     ?>
```

Как уже было описано выше, функция move_uploaded_file предназначена для загрузки файла на сервер. Зачастую при подобных операциях важно ограничивать размер загружаемого файла (ресурсы сервера не бесконечны), а также фильтровать файлы по типу (например, разрешать загружать файлы определенных типов). В следующем примере рассмотрено, как может быть организована загрузка файла на сервер, размером не более 30 000 байт (≈30 Кбайт). На лабораторных работах будет рассмотрен пример с проверкой типов загружаемых файлов.

Пусть имеется файл **index.php**, на котором есть форма для загрузки файлов. Форма должна содержать обязательный атрибут enctype="multipart/form-data" для загрузки файлов.

```
<form enctype="multipart/form-data" action="parse.php" meth-
od="post">
     <input type="hidden" name="MAX_FILE_SIZE" value="30000"
     />
     Загрузить файл:
     <input type="file" name="myfile" /><br>
     <input type="submit" value="Отправить файл" />
</form>
```

Обработчиком формы является файл **parse.php**, который должен получить файл с формы и организовать его загрузку на сервер.

```
     <?
     $uploaddir = 'c:/uploads/'; /*будем сохранять загружаемые
     файлы в эту директорию */
     $destination = $uploaddir . $_FILES['myfile']['name']; /*
     имя файла оставим неизменным */
     print "<pre>";
     if  (move_uploaded_file(  $_FILES['myfile']['tmp_name'],
     $destination)) {
          /* перемещаем файл из временной папки в выбранную
     директорию для хранения */
          print "Файл успешно загружен <br>"; }
          else { echo "Произошла ошибка при загрузке файла.
     Некоторая отладочная информация:<br>"; print_r($_FILES);
          }
     print "</pre>";
     ?>
```

Помимо работы с файлами, в PHP имеется возможность управления директориями: создание каталога и удаление каталога.

- mkdir($directory, [$permission]) – создает каталог $directory. $permission по умолчанию принимает значение 0777, что означает самые широкие права. Больше информации о правах доступа можно узнать на странице руководства функции chmod().
- rmdir($directory) – удаляет указанную директорию. Директория должна быть пустой и должны иметься необходимые для этого права. При неудачном выполнении будет сгенерирована ошибка уровня E_WARNING.

ГЛАВА 2. ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ О СИСТЕМЕ УПРАВЛЕНИЯ БАЗАМИ ДАННЫХ MYSQL

2.1. Особенности реляционной базы данных MySQL

На сегодняшний день большинство сайтов, порталов, на которые вы заходите ежедневно, хранит свой контент в базах данных. Это намного более эффективно по сравнению с созданием отдельных HTML-страниц. Любой контент, который вы видите на сайтах (пункты меню, статьи, комментарии пользователей, фотографии, списки друзей в соцсетях и многое другое), хранится в удобном, структурированном виде. Базы данных состоят из множества таблиц, которые связаны друг с другом. Запросы, которые выполняются к базам данных, отличаются лаконичностью, универсальностью (для большинства баз данных используется единый язык структурированных запросов SQL). По сравнению с обработкой файлов, SQL позволяет выполнять запросы гораздо быстрее, к тому же эффективно решается проблема одновременного доступа множества запросов к одним и тем же данным.

Приступив к изучению данного курса, студенты уже должны быть знакомы с основами работы в системах управления базами данных, например Microsoft Access или Microsoft SQLServer [16]. Восполнить пробелы или освежить знания по базам данным можно с помощью данных учебных пособий [7; 17].

MySQL – это реляционная система управления базами данных. То есть данные в ее базах хранятся в виде логически связанных между собой таблиц, доступ к которым осуществляется с помощью языка запросов SQL. MySQL – свободно распространяемая система, т.е. платить за ее применение не нужно. Кроме того, это достаточно быстрая, надежная и, главное, простая в использовании СУБД, вполне подходящая для не слишком глобальных проектов. Связка «PHP и MySQL» достаточно прочно зарекомендовала себя при разработке небольших проектов.

Данные в базе структурированы в виде таблиц, модель представления которых называется реляционной (*relation – от англ.* отношение). Автором реляционной модели является Э. Кодд, который первым предложил использовать для обработки данных аппарат теории множеств (объединение, пересечение, разность, декартово про-

изведение) и показал, что любое представление данных сводится к совокупности двумерных таблиц особого вида, известного в математике как отношение [19, с. 163]. Все таблицы в базе данных связаны между собой. Это позволяет устранить избыточность данных, дублирование.

Каждая запись в таблице должна быть уникальна, чтобы при проведении запросов обращение к записи было однозначным. Поле или набор полей, которые однозначно характеризуют запись, называют первичным ключом [17]. Ключи также используются для связи между таблицами (например, чтобы получить связные данные из таблицы о том, какой пользователь какую оценку получил за тест, не нужно хранить все эти данные в одной таблице, все организуется через отношения). Для удобства в качестве ключевого поля чаще всего задают не несколько полей, а отдельное поле, все значения которого генерируются по порядку. Таким образом, СУБД может организовать без участия разработчика уникальные идентификаторы записей. В MySQL мы чаще всего будем называть это поле id, от слова *identification* (анг.) – идентифицировать. Данное поле обозначается как первичный ключ (Primary key), и в качестве значений по умолчанию устанавливается AUTO_INCREMENT (счетчик), что означает, что каждая новая запись будет иметь последовательное значение. И даже если будут удалены какие-то записи, счетчик не собьется, а нумерация будет происходить последовательно с увеличением на единицу.

2.2. Программное обеспечение для работы с MySQL

Для удобства работы с MySQL разработано достаточно много удобных утилит, хотя с MySQL можно работать и из интерфейса обычной командной строки.

Самым распространенным решением для работы с MySQL является утилита phpMyAdmin, которая написано на PHP и является веб-приложением. PhpMyAdmin входит в состав Denwer, и для того чтобы запустить его, достаточно стартануть веб-сервер и обратиться в браузере по адресу http://localhost/tools/phpmyadmin. Однако следует понимать, что веб-приложение всегда будет работать несколько медленнее десктопных приложений. Но для наших лабораторных работ данное программное обеспечение будет вполне подходящим.

Для всех утилит по работе с MySQL характерны общие возможности, такие как:

- создание, редактирование, удаление, копирование баз данных;
- создание, редактирование, удаление, копирование таблиц;

- запуск и отладка SQL-запросов;
- просмотр содержимого баз данных и таблиц;
- экспорт / импорт баз данных, таблиц и отдельных записей;
- управление пользователями и правами пользователей к доступу по управлению базами данных;
- и др. сервисные функции.

Интерфейс phpMyAdmin представлен на рисунке 19.

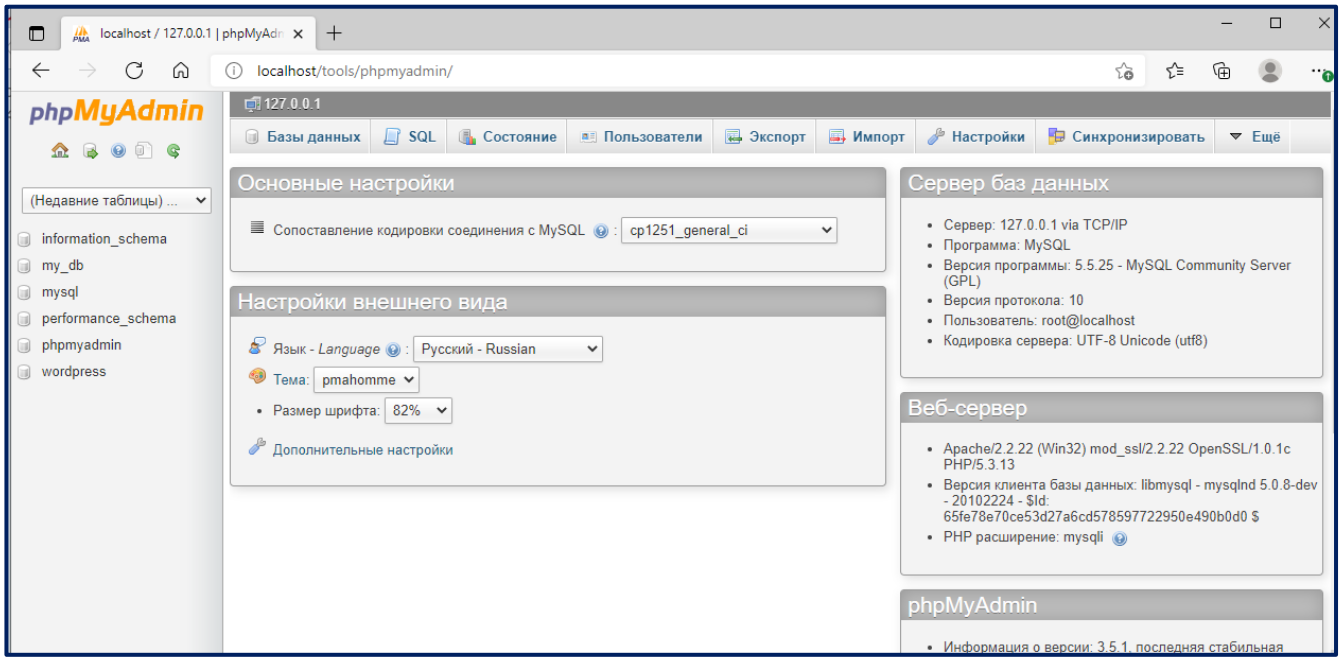


Рис. 19. Интерфейс phpMyAdmin

HeidiSQL [1] – это бесплатное, настольное (десктопное) приложение для работы с базами данных MySQL, MariaDB, Microsoft SQL, PostgreSQL и SQLite. Помимо всех базовых возможностей по работе с базами данными и таблицами, преимуществом heidiSQL является то, что ПО позволяет подключаться к различным серверам баз данных в одном окне; скорость выполнения работы значительно быстрее, чем в phpMyAdmin, работает с триггерами, процедурами SQL; есть возможность сохранения SQL-запросов; выполняет очень удобный поиск по всей таблице; позволяет легко импортировать / экспортировать данные в текстовые и табличные форматы; возможность написания запросов SQL с подсветкой синтаксиса и многое другое. Если вы всерьез заинтересовались веб-разработкой, оцените все возможности heidiSQL. Интерфейс показан на рисунке 20.

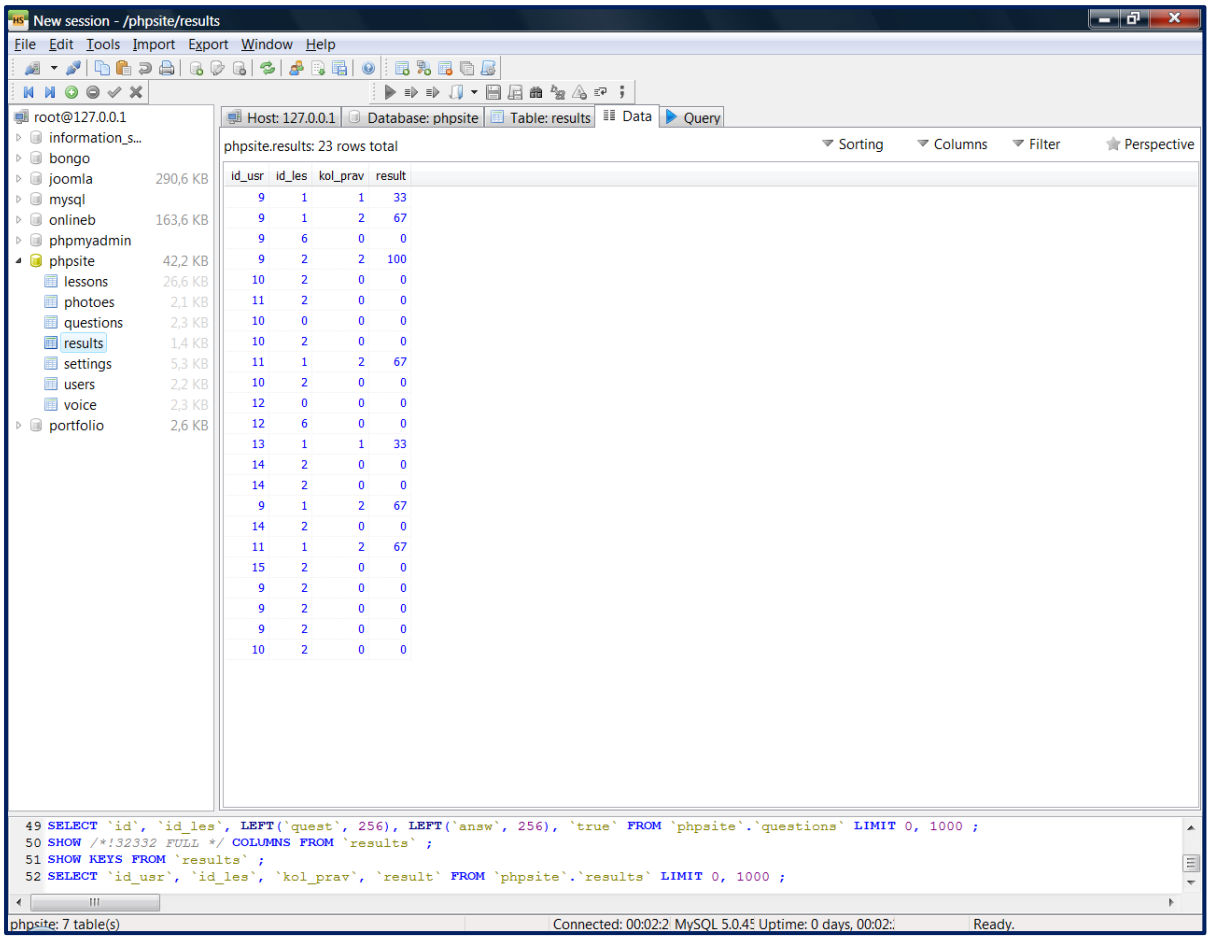


Рис. 20. Интерфейс heidiSQL

SQLYOG – когда-то начатый как студенческий проект, который на сегодняшний день является достаточно профессиональным решением. Доступна как бесплатная, так и платная версии. Как и heidiSQL, SQLYOG является настольным (десктопным) приложением, следовательно, скорость его работы будет достаточно высокой. Главные функции SQLyog:

- удобный конструктор запросов;
- умное автозавершение работы;
- интеллектуальное дополнение кода;
- туннелирование HTTP и HTTPS;
- туннелирование SSH;
- возможность синхронизации данных;
- полноценная поддержка Юникода.

Интерфейс программы представлен на рисунке 21.

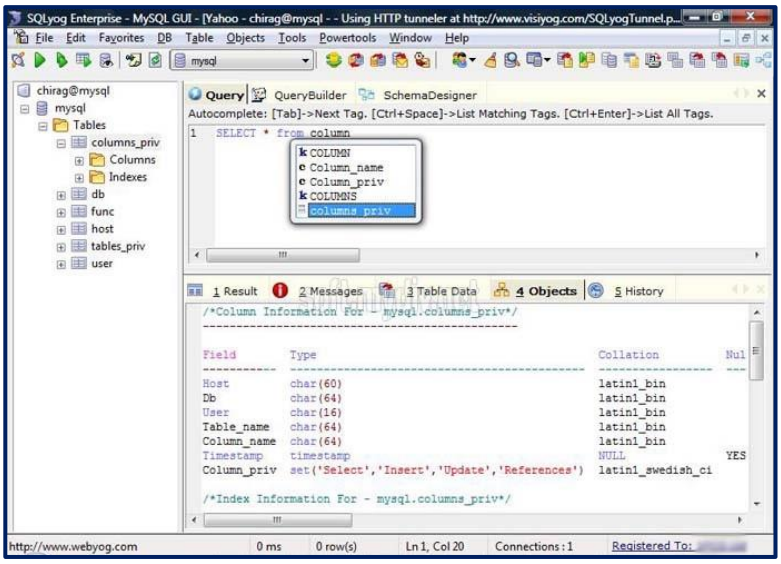


Рис. 21. Интерфейс SQLYOG

2.3. Создание базы данных, таблицы с использованием phpMyAdmin

Рассмотрим вопросы создания базы данных, таблиц баз данных с помощью утилиты phpMyAdmin. Проектирование архитектуры базы данных приложения чаще всего предшествует написанию кода на php. Действительно, важно детально погрузиться в предметную область, создать группу таблиц, заполнить их предварительными сведениями и только затем переходить к коду обработки содержимого баз данных на PHP.

Для создания базы данных достаточно указать ее название в специальное поле в phpMyAdmin по адресу http://localhost/tools/phpmyadmin/ (рисунок 22).

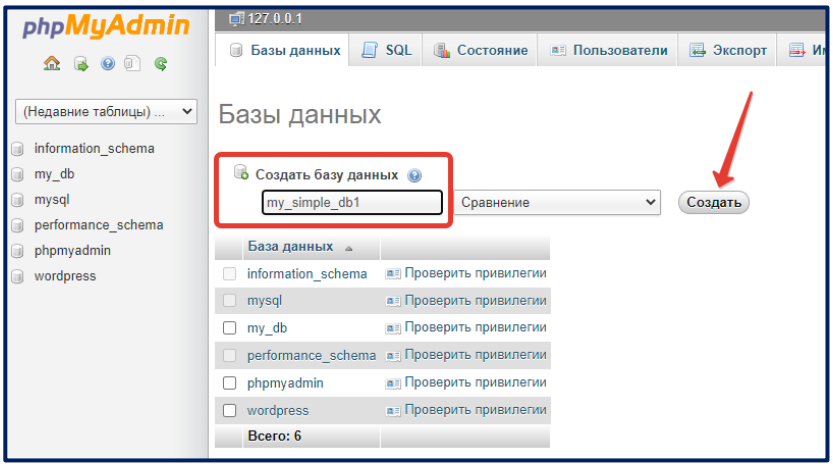


Рис. 22. Создание базы данных в phpMyAdmin

Имя базы данных может состоять из символов латинского алфавита (хотя в некоторых случаях допускается используется кириллицы, мы не рекомендуем это делать), цифр, знака доллара «$», знака нижнего подчеркивания «_». Запрещено использовать

символ пробела « », точки «.», символы слешей «\», «/». Также ограничение накладывается на длину названия базы данных – она не должна превышать 64 символа.

Фактически база данных – это только имя, сами данные содержат таблицы баз данных. После создания базы данных в структуре каталогов вашего сервера формируется папка с названием вашей базы данных (рисунок 23).

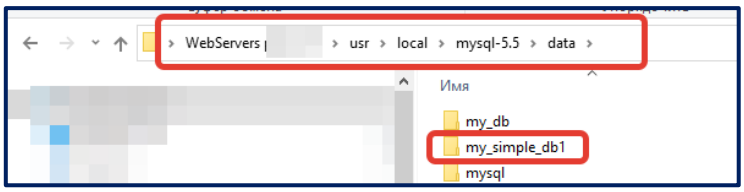


Рис. 23. Каталог созданной базы данных

Для созданной базы данных можно создать пользователя, которому будут доступны определенные привилегии по управлению базой данных. К таким привилегиям относят: управление данными в таблицах (выборка, добавление, редактирование, удаление данных); управление структурой базы данных (добавление / удаление новых таблиц, индексов, уничтожение базы данных и др.); администрирование базы данных. Создать пользователя можно на странице **Привилегии** для базы данных. Процедура не сложна: нужно придумать имя пользователя, пароль, подтвердить пароль, проверить или отметить галочкой базу данных, для которой назначаются привилегии и отметить галочкой те опции, которые пользователь сможет выполнять (рисунок 24).

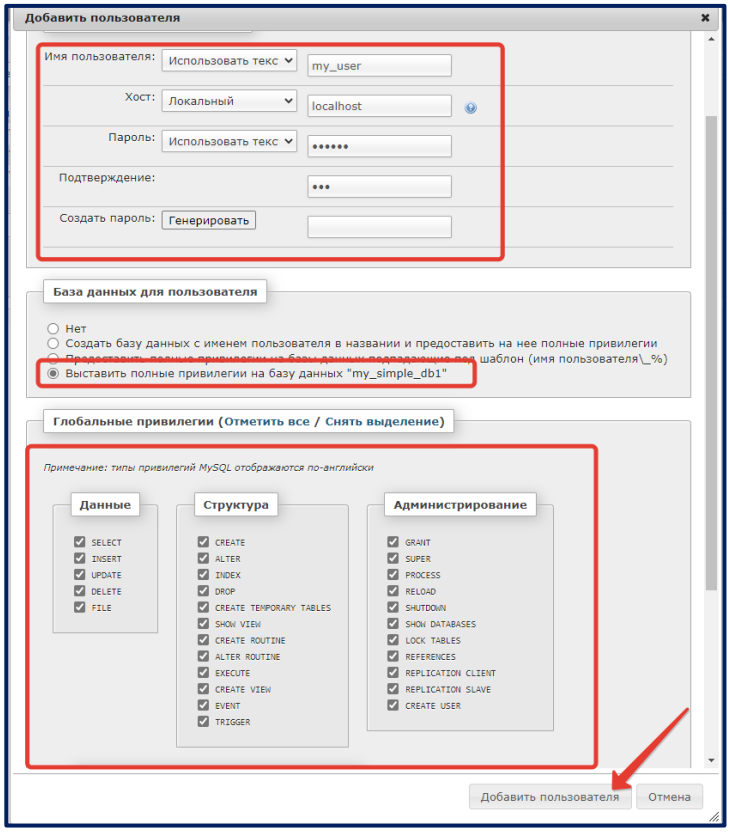


Рис. 24. Создание пользователя и определение его привилегий

После этого можно перейти к созданию таблицы в базе данных. На странице с выбранной базой данных есть поле, где можно задать имя таблицы и установить количество полей (рисунок 25). Если заранее вы не можете знать, сколько полей в таблице у вас будет, то можно поставить любое число – лишнее phpMyAdmin уберет, а если нужно будет больше полей, то их можно дополнить.

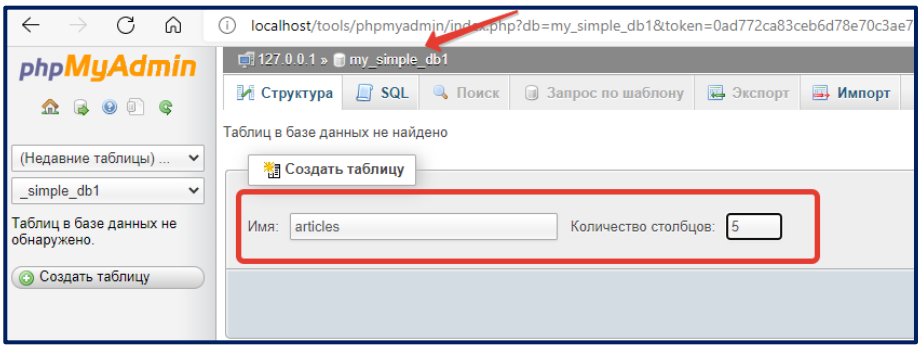


Рис. 25. Создание таблицы в базе данных

После нажатия на кнопку Ok, появится форма с созданием новых полей, указанием их типов, длины и некоторых иных параметров. Для начала рассмотрим, какие типы полей есть в MySQL. Тип поля определяет ограничения на данные и операции, которые могут быть произведены с данными этого типа. Как и в системах программирования, в MySQL тип поля определяет внутренний формат представления данных. Так, например, тип int может содержать только целочисленный набор данных, varchar – строки ограниченной длины. И очевидно, что действия, которые можно производить с целыми числами – совсем не то же самое, что действия со строками.

К целочисленным типам данных в MySQL относят:

- **TINYINT**. Диапазон значений от –127 до 128, либо 0 до 255, в зависимости от того, может ли это поле быть отрицательным.
- **SMALLINT**. Диапазон значений: –32 768 до 32 767, либо от 0 до 65 535.
- **MEDIUMINT**. Диапазон значений: от –8 388 608 до 8 388 607, либо от 0 до 16 777 215.
- **INT**. Диапазон значений: от –2 147 483 648 до 2 147 483 647, либо от 0 до 4 294 967 295.
- **BIGINT**. Диапазон значений: от –9 223 372 036 854 775 808 до 9 223 372 036 854 775 807, либо от 0 до 18 446 744 073 709 551 615.

К вещественным типам данных относят:

- **FLOAT**. Точность одинарная, то есть число знаков после запятой может быть не более 24-х (для двоичного представления). Диапазон значений: от -3,402823466E+38 до -1,175494351E-38, 0, и от 1,175494351E-38 до 3,402823466E+38.
- **DOUBLE**. Двойная точность. Количество знаков после запятой может составлять до 53-х (для двоичного представления). Допустимые значения: от -1,7976931348623157E+308 до –2,2250738585072014E–308, 0, и от 2,2250738585072014E–308 до 1,7976931348623157E+308. Данный тип используется, если нужны действительно огромные числа.
- **DECIMAL**. Это число, похожее на тип DOUBLE, но хранится оно в виде строки. И, фактически, интервал допустимых значений определяется наличием знака "–" и ".". Если эти знаки отсутствуют, то допустимый интервал такой же, как и у DOUBLE.

К строковому типу данных относят:

- **TEXT**. Максимальная длина 65 535 символов. Самый используемый вариант при хранении текстовых данных.
- **TINYTEXT**. Текст с длиной от 0 до 255 символов.
- **MEDIUMTEXT**. Текст с длиной от 0 до 16 777 215 символов.
- **LONGTEXT**. Текст с длиной от 0 до 4 294 967 295 символов.
- **VARCHAR**. Текст переменной длины от 0 до 255 символов.
- **CHAR**. Длина фиксированная (независимо от количества переданных символов). Диапазон составляет от 0 до 255 символов. При передаче данных меньше 255 символов в конце к данным дописываются пробелы, чтобы длина строки достигла заданного размера.

К типу даты / времени относят:

- **DATE**. Хранит дату. Формат следующий: YYYY-MM-DD (год, месяц, день). Например, такое значение будет удовлетворять этому полю: 2021-05-02.
- **DATETIME**. Хранит дату и время. Формат следующий: YYYY-MM-DD HH:MM:SS (год-месяц-день час-минута-секунда). Например: 2021-04-21 09:41:22
- **TIMESTAMP**. Хранит дату и время. Имеет следующие форматы: YYYYMMDDHHMMSS, YYMMDDHHMMSS, YYYYMMDD, YYMMDD.
- **TIME**. Хранит время. Формат: HH:MM:SS. Например: 09:21:55.
- **YEAR**. Хранит дату (год). Форматы: YY, YYYY.

Самыми популярными типами данных из всех перечисленных выше являются int, float, varchar, text, date, datetime.

Некоторым полям можно задавать атрибуты, например числовое поле делать автоматическим счетчиком или определять, что значения определенного поля будут неотрицательными:

- Атрибут **AUTO_INCREMENT** – генерирует новое порядковое значение для строк;
- Атрибут **UNSIGNED** – данное числовое значение будет неотрицательным.

Подробную информацию о типах данных MySQL можно получить на ресурсе [2].

Определившись с типами данных, мы можем создать таблицу. Таблица articles будет содержать сведения о публикуемых статьях (подробнее, как создать функционирующий блог, мы рассмотрим в рамках лабораторного практикума).

Первое поле по традиции назовем **id** (от слова *identify* – идентифицировать). Оно предназначено для хранения уникального номера для каждой статьи. Тип этого поля INT, длиной 11 знаков (значит хранить можно максимально 11-значное десятичное число). Также ему добавляется атрибут **AUTO_INCREMENT**. Указывается, что поле является первичным ключом **PRIMARY**.

Второе поле предназначено для хранения названия статьи, назовем его **title**. Так как заголовок статьи вряд ли будет занимать больше двух строк текста, то его тип можно определить, как VARCHAR, длиной 255 символов.

Третье поле будет хранить краткий анонс статьи (описание), можно его назвать **description**. Его можно определить типа TEXT. Данное поле не имеет ограничений.

Четвертое поле будет хранить полный текст статьи **text**. Его тип также будет TEXT.

Пятое поле будет хранить ФИО автора статьи, будет называться **author**. Тип поля можно сделать VARCHAR, длиной до 255 символов.

Создание структуры таблицы articles показано на рисунке 26.

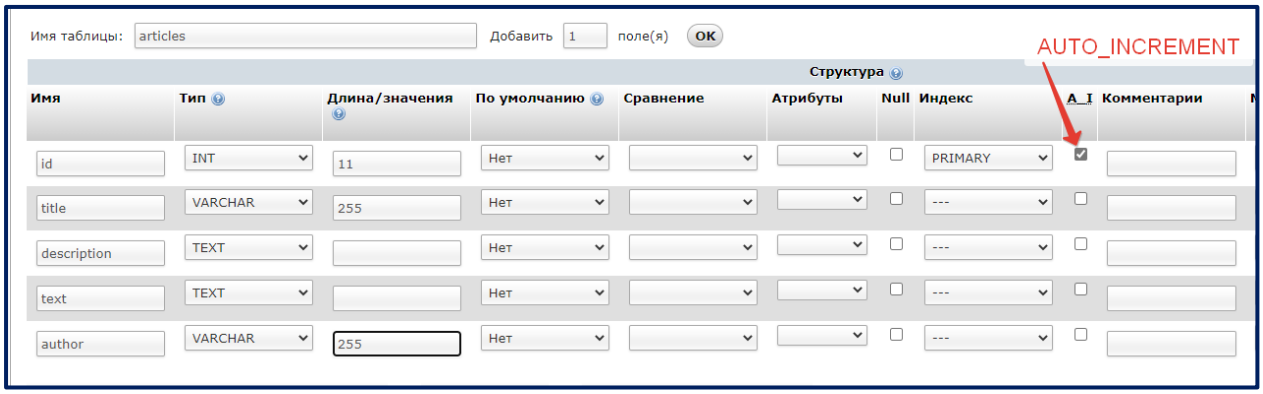


Рис. 26. Создание структуры таблицы articles

В любой момент можно внести изменения в структуру таблицы – добавить новые поля или удалить / отредактировать имеющиеся.

Мы можем добавить поле для хранения даты создания статьи. Назовем его date_created, оно будет типа DATETIME. Чтобы его внести, нужно перейти на вкладку **Структура** для нашей таблицы и поставить, после какого поля мы хотим добавить новое поле (рисунок 27). После этого поле заполняется аналогичным образом.

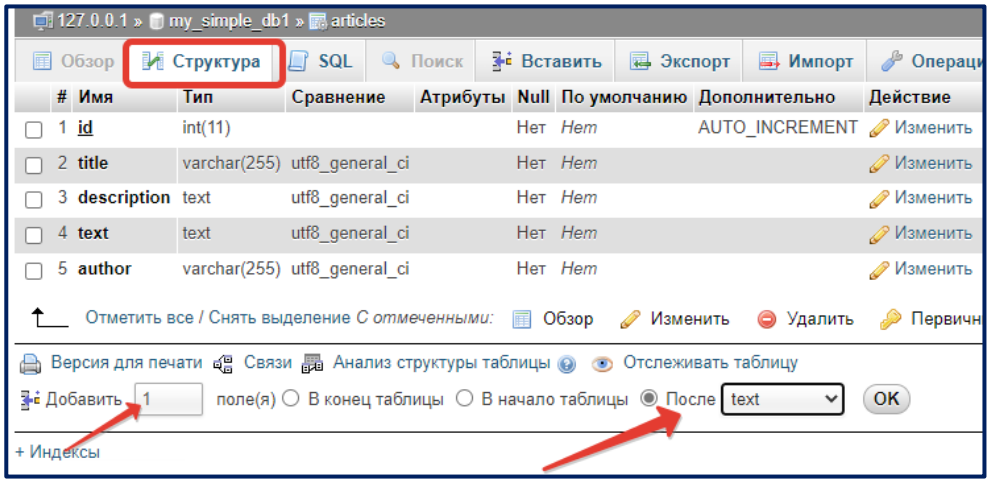


Рис. 27. Процедура добавления нового поля в таблицу

На вкладке Структура также можно любое поле отредактировать (нажать на карандаш Изменить) или удалить (нажать на Удалить).

Сейчас у нас есть таблица, но в ней нет данных. Не начинайте писать код на PHP не заполнив нужной таблицы. Как минимум две записи нужно внести в таблицу. Дело в том, что данные из таблиц будут формироваться в циклах на PHP (как это делается, мы рассмотрим позднее). Если у вас будет в таблице только одна запись, то вы не сможете протестировать работу такого цикла – одна запись может выводиться по одной выборке и без цикла. А вот если в таблице у вас две записи и при организации кода на PHP вы также получили две записи, то можно говорить, что все сделано правильно.

Чтобы вставить данные в таблицу, нужно перейти на вкладку **Вставить**. По умолчанию phpMyAdmin сразу предлагает вставить две записи. Заполняются только поля столбца **Значения**. Здесь следует отметить, что если мы установили для поля id атрибут AUTO_INCREMENT, то задавать его значения не нужно (оставляйте это поле пустым). На рисунке 28 показан процесс внесения данных одной строки таблицы articles.

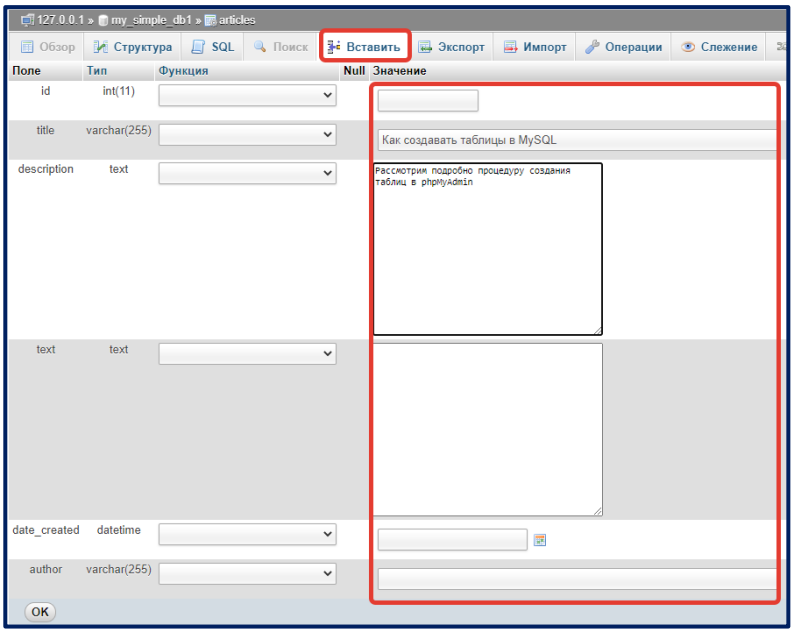


Рис. 28. Процедура внесения строки в таблицу articles

Следует отметить, что работать с phpMyAdmin просто и интуитивно понятно, если вы уже знакомы с системами управления базами данных. Многие элементы подписаны, есть подсказки, главное – быть внимательным.

2.4. Оператор SELECT

Рассмотрим наиболее популярные запросы к базам данных. Мы не будем углубляться очень детально в специфику SQL-запросов, можно их изучить с помощью учебного пособия [17; 20], документации [2].

Запрос на выборку SELECT позволяет выбрать данные из таблицы с учетом некоторого условия.

Общи вид запроса на выборку следующий:

```
SELECT <что_выбираем>
FROM <имя_таблицы>
[WHERE <условие_для_выборки>]
[ORDER BY <название_поля> [ASC | DESC]]
[LIMIT [offset,] rows]
]
```

<что_выбираем> может быть перечнем полей, либо оператором *, который заменяет выбор всех полей.

Сравните:

```
SELECT id, title, description, text, author FROM articles
```

или

```
SELECT * FROM articles
```

Выражения **WHERE**, **ORDER BY** и **LIMIT** не являются обязательными в разделе. Именно поэтому в нашем запросе они заключены в квадратные скобки.

Выражение **WHERE** позволяет задать условие выборки. Если нужно составить сложное условие, то могут быть использованы операторы **AND** (логическое «И»), **OR** (логическое «ИЛИ») и **NOT** (логическое «НЕ»).

Пример, запрос на выборку всех записей со значением поля id, равное 2, и поля author, содержащего 'Расмус Лердорф':

```
SELECT * FROM articles WHERE id = 2 AND author='Расмус Лердорф'
```

MySQL условие **LIKE** позволяет использовать шаблоны в операторе **WHERE**. Это позволяет выполнять сопоставление шаблонов. В сочетании с оператором **LIKE** используются два подстановочных знака:

- «%» – знак процента представляет ноль, один или несколько символов;
- «_» – знак подчеркивания представляет один символ.

Примеры работы с операторами представления находятся в таблице 3.

Таблица 3

Примеры операторов представления

LIKE Оператор	Описание
WHERE FieldName LIKE 'a%'	Находит любые значения, которые начинаются с «a»
WHERE FieldName LIKE '%a'	Находит любые значения, которые заканчиваются «a»
WHERE FieldName LIKE '%or%'	Находит любые значения, которые имеют значение «or» в любом положении
WHERE FieldName LIKE '_r%'	Находит все значения, которые имеют значение «r» во второй позиции
WHERE FieldName LIKE 'a_%_%'	Находит любые значения, которые начинаются с «a» и длиной не менее 3 символов
WHERE FieldName LIKE 'a%o'	Находит любые значения, которые начинаются с «a» и заканчиваются на "o"

Выражение **ORDER BY** используется для сортировки записей в вашем результирующем наборе. Здесь обязательно указывается поле для сортировки и тип сортировки: по возрастанию (**ASC**) и по убыванию (**DESC**). Если атрибуты **ASC** или **DESC** не указаны в операторе **ORDER BY**, результаты будут отсортированы по полю в порядке возрастания. Это эквивалентно выражению **ORDER BY <поле> ASC**.

Например, следующий запрос отсортирует сначала в порядке убывания по автору, затем по названию в порядке возрастания:

```
SELECT * FROM articles ORDER BY author DESC, title
```

Выражение **LIMIT** может использоваться для ограничения количества строк, возвращенных командой **SELECT**. **LIMIT** принимает один или два числовых аргумента. Эти аргументы должны быть целочисленными константами. Если заданы два аргумента, то первый указывает на начало первой возвращаемой строки, а второй задает максимальное количество возвращаемых строк. При этом смещение начальной строки равно 0 (не 1).

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

возвращает строки 6–15.

Часто возникает необходимость выбрать из базы данных некоторые строки в случайном порядке и вывести только часть из них. Например, на сайте в фотогалерее требуется вывести 5 случайных фотографий. Для этого MySQL имеет встроенную функцию RAND(). Работает она следующим образом:

```
SELECT * FROM articles ORDER BY RAND() LIMIT 5
```

Здесь функция RAND() сгенерирует случайное число для каждой строки в таблице. Предложение ORDER BY сортирует все строки в таблице по случайному числу, сгенерированному функцией RAND(). Предложение LIMIT выбирает первые 5 строк в наборе результатов, отсортированных случайным образом. Если LIMIT не указан, то будут выведены все записи в случайном порядке. Данный способ окажется гораздо быстрее по скорости и лаконичнее по записи кода, чем если бы мы доверили ту же задачу PHP.

Если требуется отсортировать записи в некотором заранее известном порядке для разработчика (сортировка по определенной последовательности), то применяется функция FIELD(). В данной функции на первом месте указывается поле, по которому производится сортировка, затем через запятую перечисляется порядок, в котором должны выводиться записи по указанному полю.

```
SELECT * FROM articles ORDER BY FIELD(id, 4,2,1,3)
```

Данный запрос выведет первые записи из таблицы **articles** в порядке поля **id – 4, 2, 1, 3** записи. Если в таблице есть другие записи, то заданная упорядоченность работать не будет. Если же значений в функции FIELD() больше, чем строк в таблице, то заданный порядок сохранится.

К запросу на выборку применимы также и некоторые агрегатные функции. К ним относят подсчет суммы, количества, минимального значения, максимального значения, среднего арифметического. Агрегатные функции SQL действуют в отношении значений столбца с целью получения единого результирующего значения. Проще говоря, используя эти функции, мы можем из множества значений столбца получить одно единственное. Такие же операции программист может сделать и с использованием языка программирования, однако MySQL выполняет подобные запросы на порядок быстрее.

Общий формат обращения к агрегатной функции в операторе SELECT следующий:

```
SELECT <функция>(<поле>) FROM <таблица> [WHERE …]
```

В качестве функций может выступать одно из следующих значений

- AVG – вычисляет среднее значение;
- SUM – вычисляет сумму значений;
- MIN – вычисляет наименьшее значение;
- MAX – вычисляет наибольшее значение;
- COUNT – вычисляет количество строк в запросе.

Пусть имеется таблица со списком статей articles. Чтобы подсчитать общее количество статей в таблице, можно написать запрос:

```
SELECT COUNT(id) FROM articles
```

В данном случае запрос вернет нам количество записей из таблицы articles. Можно записать этот запрос несколько иначе – вместо поля id использовать символ * (все поля). Результат получится одинаковым.

Пусть имеется таблица guest_book, хранящая в полях id, name, message, email, date_created информацию о сообщениях гостевой книги (подобную таблицу мы сделаем позже в рамках лабораторных работ). Проиллюстрируем запрос, который выведет даты самого раннего и самого позднего сообщений:

```
SELECT MIN(date_created),MAX(date_created) FROM guest_book
```

При проектировании баз данных, каждая таблица относится к описанию одной сущности. Большой задачей является нормализация данных при представлении в таблицах. В результате проектирования часто возникает необходимость отдельные сущности хранить в разных таблицах, но между таблицами при этом существуют связи. Простым примером здесь могут служить сущность «Студенты» (ФИО, дата рождения, принадлежность группе) и сущность «Группы» (номер группы, факультет). Связь устанавливается через номер группы – понятно, что каждый студент учится в какой-то группе, которая в свою очередь находится на определенном факультете. Но в самой таблице «Студенты» вовсе не нужно для каждого студента прописывать принадлежность к факультету, достаточно установить связь, к какой группе он относится, и потом уже с помощью запросов особого типа получить сведения по группе. Подробную информацию о нормализации данных при проектировании реляционных таблиц можно получить из учебного пособия А.А. Рузакова «Управление данными» [17, с. 32].

Связи между таблицами MySQL выполняются в результате запроса JOIN. Пусть имеются две таблицы:

Students (id, fio, birthday, groupid);

Groups (id, name, faculty).

Связь устанавливается через соединение поля groupid таблицы Students и поля id таблицы Groups. Задача запроса – найти одинаковые значения в данных полях в обеих таблицах. Чтобы это произошло, значения id в таблице Groups должны быть уникальными, а вот значения groupid могут повторяться.

Таблица 4

Таблица Students

id	fio	birthday	groupid
1	Иванов Иван Иванович	01.01.1990	1
2	Петров Петр Петрович	02.02.1990	2
3	Сидорова Мария Владимировна	03.03.1989	1
4	Никитина Елена Викторовна	04.04.1990	3

Таблица 5

Таблица Groups

id	name	faculty
1	213-092-5-1	Математики, физики, информатики
2	409-053-5-1	Филологический
3	302-041-5-1	Исторический
4	511-066-5-1	Иностранных языков

Чтобы вывести, в какой группе и на каком факультете учится каждый студент, запрос MySQL будет выглядеть следующим образом:

```
SELECT * FROM Students, Groups WHERE Students.groupid = Groups.id
```

В результате запроса получим следующие сведения (рисунок 29).

+ Параметры

id	fio	birthday	groupid	id	name	faculty
1	Иванов Иван Иванович	1990-01-01	1	1	213-092-5-1	Математики, физики, информатики
3	Сидорова Мария Владимировна	1989-03-03	1	1	213-092-5-1	Математики, физики, информатики
2	Петров Петр Петрович	1990-02-02	2	2	409-053-5-1	Филологический
4	Никитина Елена Викторовна	1990-04-04	3	3	302-041-5-1	Исторический

Рис. 29. Результат запроса на соединение

Алгоритм здесь несложный: берётся первая запись из таблицы Students. Далее берётся её id и анализируются все записи из таблицы Groups, добавляя в результат те, у которых groupid равен id из таблицы Groups. Таким образом на первой итерации собираются все сведения у первого студента. На второй итерации собираются все сведения у второго студента и так далее.

Оператор JOIN выполняет полное объединение. Операторы JOIN, INNER JOIN и «,» (запятая) с указанием условий объединения дают пересечение для всех совпадающих значений из условия.

```
SELECT t1.*, t2.* FROM t1, t2 ON t1.i1 = t2.i2
```

Аналогично можно использовать и такую запись

```
SELECT t1.*, t2.* FROM t1 INNER JOIN t2 ON t1.i1 = t2.i2
```

Операторы LEFT JOIN, RIGHT JOIN выполняют, соответственно, левое и правое объединения таблиц.

При использовании оператора LEFT JOIN выбираются все значения из левой таблицы (первой указанной), после чего к ним подбираются соответствующие значения из правой, если таких нет, то все значения для правой таблицы в этой строке равны NULL.

```
SELECT t1.*, t2.* FROM t1 LEFT JOIN t2 ON t1.i1 = t2.i2
```

RIGHT JOIN противоположен LEFT JOIN: всё абсолютно также, но выбираются все значения для правой таблицы из выражения.

При указании любого из операторов JOIN можно использовать ключевое слово USING() вместо ON, если названия столбцов в обеих таблицах одинаковы.

2.5. Оператор INSERT

Запрос на вставку **INSERT** позволяет вставить новые записи в таблицу. Общий вид запроса:

```
INSERT INTO <имя_таблицы>
(<поле1>, <поле2>, ... )
VALUES (<выражение1>, <выражение2>, ... )
```

Например,

```
INSERT INTO articles (id, title) VALUES (3, 'Синтаксис запроса INSERT')
```

вставит в таблицу **articles** строку с **id** равным **3** и заголовком «Синтаксис запроса INSERT». Остальные поля будут заданы значениями по умолчанию (чаще всего это NULL, 0 или пустая строка).

2.6. Оператор UPDATE

Запросы на обновление имеющихся записей **UPDATE**. Его синтаксис следующий

```
UPDATE <имя_таблицы>
SET <поле1> = <выражение1>, <поле2> = <выражение2>,
... [WHERE <условие_для_выборки>]
```

Здесь выражение **WHERE** не является обязательным, но если не задуматься над условием для выборки, то обновятся все записи в таблице.

Например, запрос

```
UPDATE articles SET date_created='2021-15-05 00:00:00', author='Oracle corporation' WHERE id = 1
```

обновит дату создания и имя автора для записи с **id = 1**. Не указанные в запросе поля останутся без изменения.

Заметим, что оператор UPDATE является небезопасным запросом, поэтому при его выполнении в последних версиях phpMyAdmin появится предупреждение. Однако необходимо всегда помнить, что без оператора WHERE будут заменены все записи. Лучше перед проведением таких запросов сделать копию таблицы.

2.7. Оператор DELETE

Запрос на удаление записей **DELETE** в общем виде выглядит так

```
DELETE FROM <имя_таблицы> [WHERE <условие_для_выборки>]
```

Например, удалит записи о статьях, созданные позже 15 мая 2021 года, запрос

```
DELETE FROM articles WHERE date_created > '2021-05-15'
```

Оператор DELETE так же, как и UPDATE, является небезопасным запросом, поэтому при его выполнении в последних версиях phpMyAdmin появится предупреждение. Однако необходимо всегда помнить, что без оператора WHERE будут удалены все записи. Лучше перед проведением таких запросов сделать копию таблицы.

2.8. Тестирование запросов в phpMyAdmin

С точки зрения PHP, запрос к базе данных на языке SQL – это обычная строка. Интерпретатор PHP не способен проверить правильность или неправильность написания запроса. Единственный тип ошибки, который PHP может выдать, если вы неправильно составили запрос, – это предупреждение, что в результате запроса PHP не получил никаких данных. При этом информации о том, что было записано в запросе неверно, у вас не будет. Прежде чем формулировать запрос непосредственно в PHP, протестируйте его на правильность выполнения через панель SQL-запросов самого MySQL-сервера. Это в первую очередь касается «безопасных» запросов SELECT, которые не меняют содержимое таблиц базы данных. Рассмотрим, как это можно сделать с помощью phpMyAdmin.

Предположим, что у нас создана база данных my_db, в которой есть таблица со статьями для нашего блога **articles.** Для таблицы **articles** были созданы поля **id** (первичный ключ, уникальный идентификатор статьи), **title** (заголовок статьи), **description** (краткая аннотация статьи), **text** (полный текст статьи), **date_created** (дата создания статьи), **author** (автор статьи). Попробуем составить и протестировать запрос на получение всех записей из таблицы **articles**.

```
SELECT * FROM articles
```

Чтобы выполнить этот запрос, нужно перейти в phpMyAdmin по адресу http://localhost/tools/phpMyAdmin, затем выбрать нужную базу данных – my_db, после этого перейти на вкладку SQL и вставить свой запрос (рисунок) и нажать на кнопку **OK** (рисунок 30).

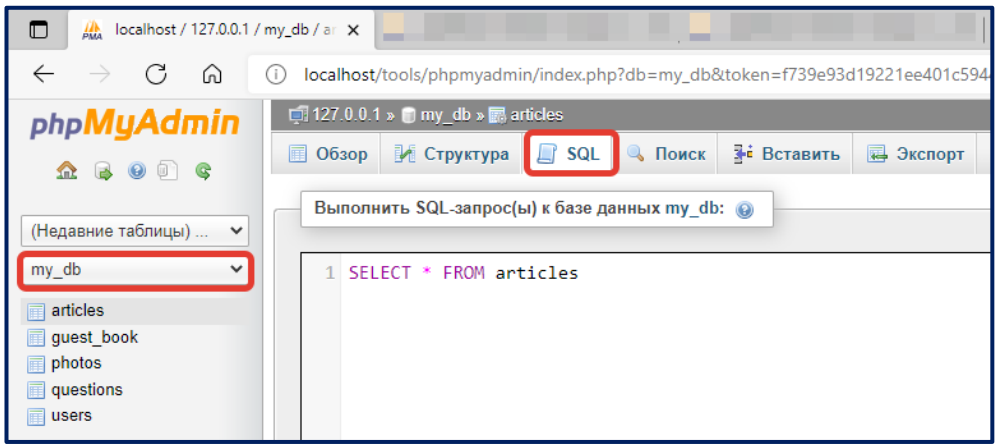


Рис. 30. Тестирование запроса SQL в панели phpMyAdmin

Настоятельно рекомендуем вам сохранять свои запросы в каком-нибудь текстовом документе (подойдет Блокнот, Notepad++), потому что выполненный запрос система не сохраняет.

Если запрос был выполнен успешно, то вы увидите таблицу со значениями (рисунок 31).

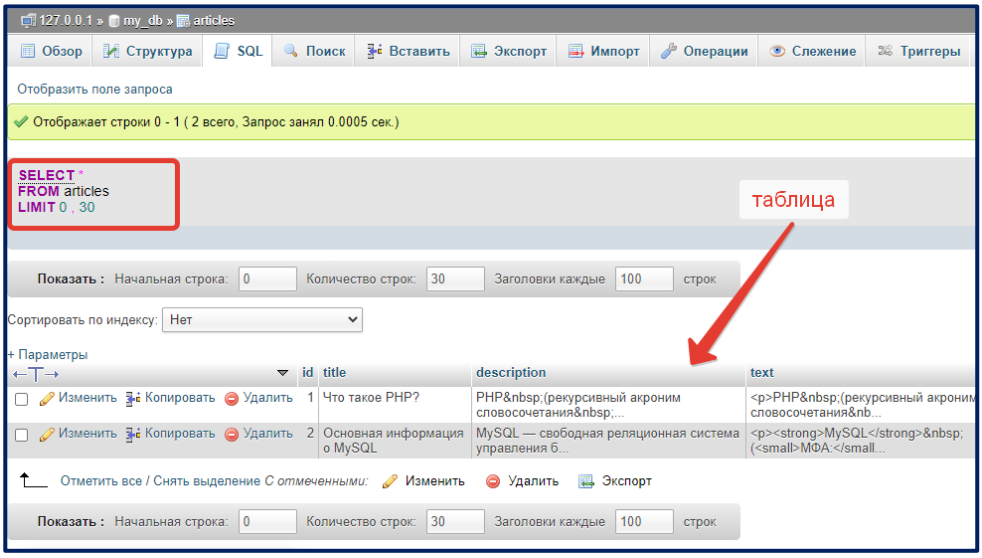


Рис. 31. Результат успешно выполненного запроса

Если же в запросе была допущена ошибка, то вы увидите сообщение, в котором несложно догадаться, где вы допустили ошибку. В результате выполнения запроса на рисунке 32 была допущена ошибка при наименовании поля. Правильно было бы обратиться к полю **author,** но в запросе обращение идет к полю **avtor**. phpMyAdmin сразу же выдал ошибку насчет этого. Правильный запрос должен выглядеть так:

```
SELECT id, title, author FROM articles
```

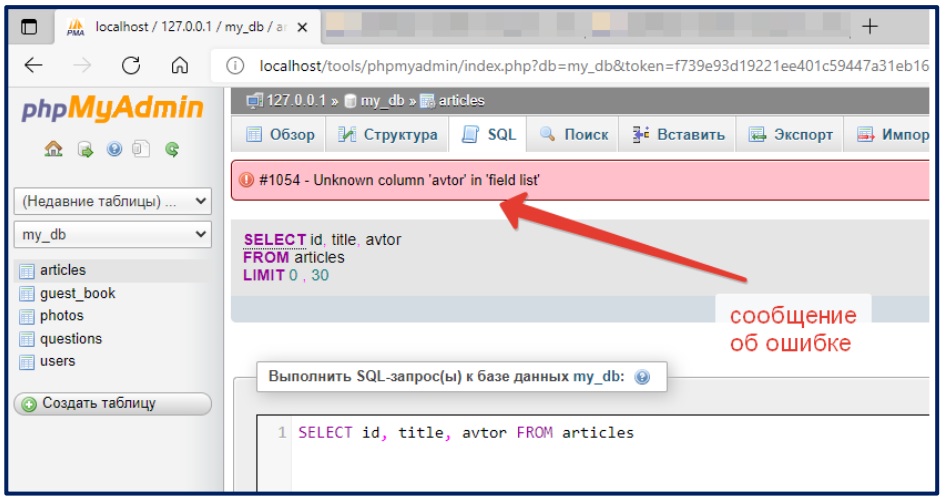


Рис. 32. Результат выполнения запроса с ошибкой

Аналогичным способом можно тестировать запросы на INSERT, UPDATE, DELETE, только нужно всегда помнить, что эти запросы меняют состояние таблиц. Если ошибочно написать в запросе UPDATE или DELETE без условия WHERE, вы рискуете изменить свои таблицы без возможности вернуть все назад. Поэтому важно перед проведением этих типов запросов сделать копию (бекап) нужной вам таблицы.

Сделать копию таблицы можно на вкладке **Операции**. Предварительно вы должны выбрать базу данных, таблицу, которую хотите скопировать. На вкладке **Операции** находится блок, в котором можно указать, как должна называться новая таблица, а также указать, что необходимо перенести при копировании: только структуру, структуру и данные или только данные. Ход копирования показан на рисунке 33.

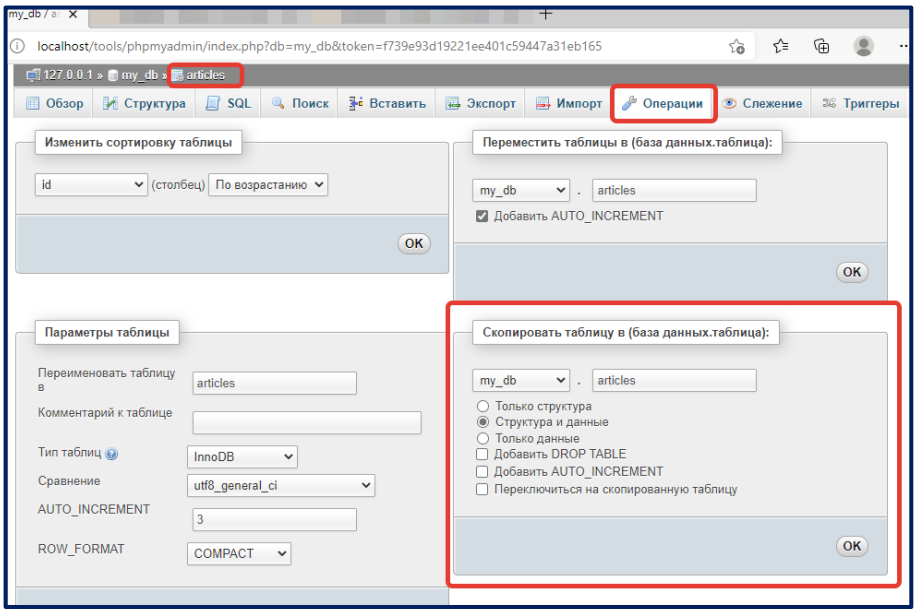


Рис. 33. Ход выполнения копирования таблицы в phpMyAdmin

2.9. Организация взаимодействия PHP и MySQL

После установки и настройки базы данных MySQL приступим к рассмотрению вопроса, каким образом PHP-скрипты соединяются с базой данных, каким образом происходит обращение к нужной таблице и как обрабатываются результаты, полученные из базы данных.

Самая первая функция, изучаемая нами в данном параграфе, производит инициализацию подключения к базе данных MySQL. Функция называется **mysql_connect ($host, $user, $password)**. Данной функции на вход подается три параметра:

- **$host** – имя сервера, на котором находится база данных, в большинстве случаев используется локальный хост **localhost**;
- **$user** – имя пользователя, которого мы создавали для подключения к базе данных. Можно использовать **root** в качестве универсального пользователя, у которого нет пароля. Но делать это можно только в период тестирования.
- **$password** – пароль пользователя, который задавался при создании базы данных.

Результат функции **mysql_connect** сохраняется в некоторую переменную, например, **$db**, которая впоследствии может быть использована в запросах.

Пример:

```
$db=mysql_connect("localhost","ivan","123456");
```

или для пользователя по умолчанию:

```
$db=mysql_connect("localhost","root","");
```

Функция **mysql_close([$db_identifier])** закрывает соединение с сервером MySQL. Возвращает TRUE, в случае успешного завершения, или FALSE, в случае возникновения ошибки. Использование **mysql_close()** не обязательно для непостоянных соединений (они автоматически закрываются в конце скрипта).

Корректная установка соединения может выглядеть следующим образом:

```
<?
$db = mysql_connect("localhost","login","password");
If (!$db) {
exit("Невозможно установить соединение: ". mysql_error());}
else
{echo "Соединение установлено";
mysql_close($db);}
?>
```

После установки соединения происходит выбор базы данных. **mysql_select_db($database_name, [$db_identifier])** выбирает для работы указанную базу данных **$database_name** на сервере, на который ссылается переданный указатель **$db_identifier**. Если параметр указателя опущен, используется последнее открытое соединение. Если нет ни

одного открытого соединения, функция попытается соединиться с сервером аналогично функции **mysql_connect()**, вызванной без параметров. Пример:

```
mysql_select_db("my_db",$db);
```

Запрос к базе данных, поступающий из сценария PHP, по сути представляет собой команду MySQL, заключенную в функцию **mysql_query()**. Именно данная функция позволяет организовать основные операторы SQL – SELECT, INSERT, UPDATE, DELETE. Общий вид функции: **mysql_query($query, [$db_identifier])**, где **$query** – это строка с запросом MySQL, **$db_identifier** – указатель на соединение с базой данных. Если указатель не указан, то используется последнее открытое соединение. Только для запросов SELECT **mysql_query()** возвращает указатель на результат запроса или FALSE, если запрос не был выполнен. В остальных случаях (INSERT, UPDATE, DELETE), mysql_query() возвращает TRUE, в случае успешного запроса, и FALSE, в случае ошибки. Значение не равное FALSE говорит о том, что запрос был выполнен успешно. О количестве затронутых или возвращённых рядов запрос не сообщает. Вполне возможна ситуация, когда успешный запрос не затронет ни одного ряда.

Пример обработки запроса – результат помещается в переменную $result для дальнейшей обработки:

```
$result = mysql_query("SELECT * FROM articles", $db);
```

В некоторых случаях, особенно когда добиться правильного написания sql-запроса сразу не удается, рекомендуется оформлять строку с запросом в виде отдельной переменной, например $query. В этом случае всегда можно распечатать запрос и протестировать его в панели phpMyAdmin.

```
$query = "SELECT id, title, author FROM articles";
print_r($query);
$result = mysql_query($query);
```

После получения информации о выполнении запроса необходимо обработать данные на стороне PHP. При этом удобно, когда данные, полученные по запросу, оформляются в виде массива.

Функция **mysql_fetch_row** – возвращает строку (результат запроса) в виде массива с числовым индексом.

Функция **mysql_fetch_array** – возвращает строку (результат запроса) в виде ассоциативного массива. Ключевыми индексами такого массива являются названия полей в таблице mysql.

Функция **mysql_result** – возвращает один элемент массива данных. Подходит для обработки агрегатных запросов.

Обработка всех данных, полученных по запросу, с использованием функций mysql_fetch_row() и mysql_fetch_array() осуществляется в циклических конструкциях. Можно использовать цикл while, do-while или for для этих целей. Далее приведем пример обработки запроса с помощью цикла while.

```
<?php
     $sql = "SELECT * FROM articles";
     $result = mysql_query($sql);
     while ($myrow = mysql_fetch_array($result)) {
          echo $myrow["id"] . " " . $myrow["title"] . " ".
          $myrow["author"] . "<br>";
          echo $myrow["desription"] . "<br>";
     }
?>
```

Если планируете использовать цикл **for**, то важно понимать, как можно заранее получить общее количество записей по запросу. Сделать это можно с помощью функции **mysql_num_rows**().

Пример использования

```
<?
$db=mysql_connect("localhost","root","");
mysql_select_db("my_db",$db);
$sql = "select id, title, description, author, date_created
from articles";
$result=mysql_query($sql,$db);
$count = mysql_num_rows($result);
For ($i=0; $i<=$count; $i++) {
     $myrow=mysql_fetch_array($result);
     echo $myrow["id"] . " " . $myrow["title"] ." ". $my-
     row["author"] . "<br>";
     echo $myrow["desription"]."<br>";
}
?>
```

Для удобства разработчиков были введены Функции возврата сообщений об ошибках из базы данных.

- **mysql_error([$db_identifier])** – возвращает текст ошибки последней операции с MySQL.
- **mysql_errno([$db_identifier])** – численное значение сообщения об ошибках от предыдущей операции MySQL. По этому коду можно расшифровать тип возникающей ошибки. Так, код ошибки 1049 сообщает, что имя базы данных неизвестно (возможно, при написании кода была допущена ошибка в имени или на данном сервере в принципе нет такой базы данных); код 1146 – указанной таблицы не найдено в базе данных.

```
<?php
$db = mysql_connect("localhost", "root", "");
mysql_select_db("abcdefg", $db);
echo mysql_errno($db) . ": " . mysql_error($db). "<br>";

mysql_select_db("my_db", $db);
mysql_query("SELECT * FROM abcdefg", $db);
echo mysql_errno($db) . ": " . mysql_error($db) . "<br>";
?>
```

Результатом выполнения данного примера будет что-то подобное:

```
1049: Unknown database 'abcdefg'
1146: Table 'my_db.abcdefg' doesn't exist
```

Другие коды ошибок можно посмотреть на ресурсе [4].

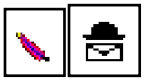
ГЛАВА 3. ЛАБОРАТОРНЫЙ ПРАКТИКУМ

3.1. Установка и настройка сервера

Задачи лабораторной работы

- установить веб-сервер Apache и сервер баз данных MySQL из набора дистрибутивов Denwer;
- написать простейшую программу.

Задание 1. Установка и настройка программного сервера.

1. Скачайте архив WebServers.zip с портала университета и распакуйте его на рабочий стол. Архив можно взять по ссылке https://cloud.mail.ru/public/KuNS/NWuy3w1eJ.
2. Для запуска, перезапуска и остановки сервера используйте программы:
   - Запуск сервера – **WebServers/denwer/run.exe;**
   - Перезапуск сервера – **WebServers/denwer/restart.exe;**
   - Остановка сервера – **WebServers/denwer/stop.exe.**
3. Запустите сервер. Если в момент запуска у вас появится окно с вводом прав администратора – проигнорируйте его (нажмите «отмена»)
4. По окончании загрузки сервера убедитесь, что появились ярлыки 

Задание 2. Создание домена для работы на локальном хосте (localhost).

1. В каталоге WebServers\home\localhost\www создайте папку **programs**. В этой папке будут храниться написанные программы.
2. Запустите сервер **WebServers/denwer/run.exe** (если сервер уже был запущен, то перезапустите его **WebServers/denwer/restart.exe**).
3. Создайте в любом текстовом редакторе (например, Блокнот, Notepad++) PHP-файл, сохраните в папку **WebServers\home\localhost\www\programs** и назовите его **1.php** (рис. 34).

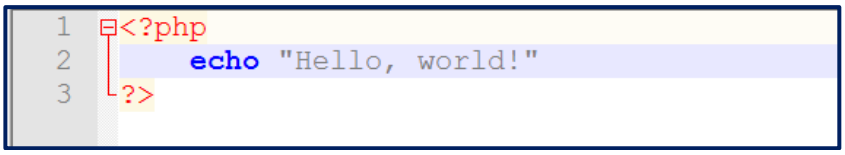


Рис. 34. Код первой программы

4. В браузере наберите адрес http://localhost/programs/1.php. Если вы увидели фразу «Hello, world!», то можно поздравить – вы все сделали верно, и ваши php программы будут работать.

Задание 3. По окончании занятия остановите сервер, запустив на выполнение программу **WebServers/denwer/stop.exe**.

Контрольные вопросы

1. Что понимается под веб-сервером? Какие функции выполняет веб-сервер?
2. Что такое хост?
3. Какие дистрибутивы входят в состав пакета Denwer?
4. Какими способами можно создать домены на веб-сервере?
5. Как узнать, запущен сервер или нет?

3.2. Создание шаблона страницы PHP-сайта

Задачи лабораторной работы

- Разметить сайт, т.е. создать его шаблон в виде таблицы;
- Оформить графически заголовок сайта и строку подписи;
- Создать меню сайта;
- Заполнить стартовую страницу сайта контентом.

Задание 1. Создадим шаблон нашей главной страницы по следующей схеме.

1. Создайте папку в каталоге **Z:\home\localhost\www**, в которой будут храниться файлы сайта (например, папка **mysite**).

2. С помощью текстового редактора Notepad++ создайте файл index.php в папке с вашим сайтом.

3. Проверьте кодировку страницы: в Notepad++ **Кодировки / Преобразовать в ANSI** (или **Encoding / Convert to ANSI**) – рисунок 35.

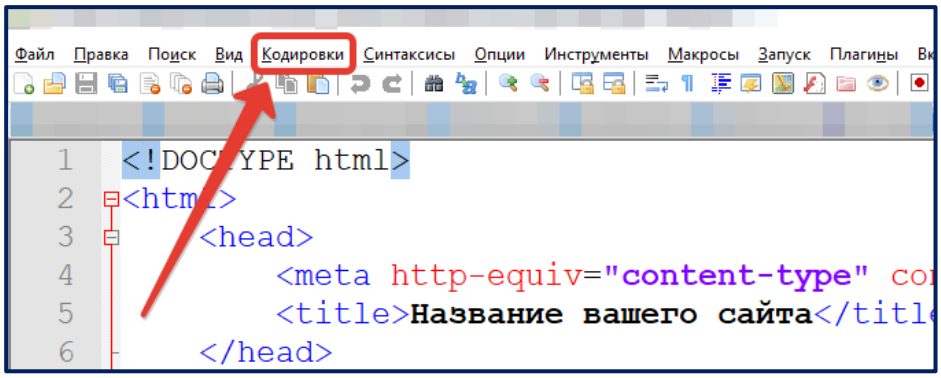
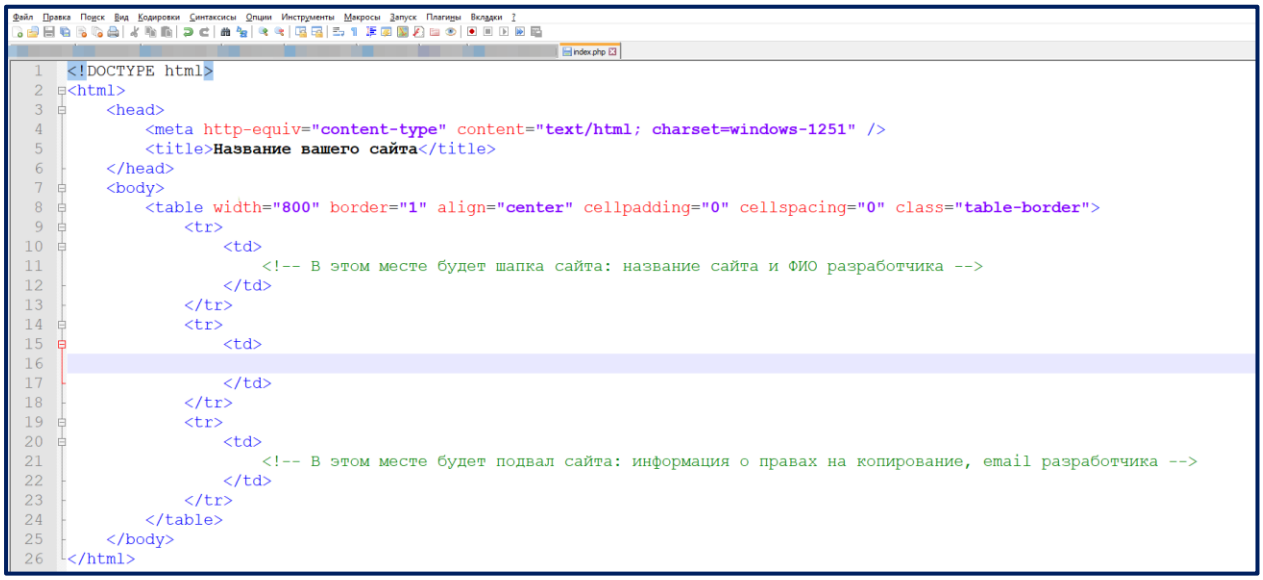


Рис. 35. Установка кодировки страницы

4. Разместите в вашем файле html-разметку страницы. В теле страницы у нас будет пока один элемент – таблица. Первая ячейка таблицы – «шапка» сайта, нижняя – «подвал», средняя – содержание сайта (контент). Обратите внимание, что HTML – язык иерархический и важно формировать отступы в коде, чтобы в дальнейшем не запутаться. Сделать это можно с помощью клавиши **TAB** (рисунок 36).



```
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="content-type" content="text/html; charset=windows-1251" />
        <title>Название вашего сайта</title>
    </head>
    <body>
        <table width="800" border="1" align="center" cellpadding="0" cellspacing="0" class="table-border">
            <tr>
                <td>
                    <!-- В этом месте будет шапка сайта: название сайта и ФИО разработчика -->
                </td>
            </tr>
            <tr>
                <td>

                </td>
            </tr>
            <tr>
                <td>
                    <!-- В этом месте будет подвал сайта: информация о правах на копирование, email разработчика -->
                </td>
            </tr>
        </table>
    </body>
</html>
```

Рис. 36. Структура HTML-кода с отступами

Пояснение

Базовая разметка html-страницы содержит тег **<html> … </html>**. Внутри тега <html> допустимо использовать только теги **<head>…</head>** («голова») и **<body> … </body>** («тело»). Тег **<title>…</title>** окружает текст, который будет выводиться в заголовке страницы (название вкладки браузера). Он используется только внутри тега **<head>…</head>**.

Как было отмечено, ключевым элементом разметки является таблица, которая ограничена HTML-тегами **<table> … </table>**. В качестве атрибутов таблицы указано:

width="800" – ширина таблицы в 800 пикселей;

border="1" – задана граница таблицы в 1 пиксель;

align="center" – выравнивание таблицы по центру;

cellpadding="0" – обнуление внутреннего отступа в ячейках таблицы (от границы до внутреннего текста);

cellspacing="0" – обнуление отступа между соседними ячейками в таблице;

class="table-border" – подключение будущего CSS-класса для последующего изменения дизайна таблицы (стили мы подключим позже).

Таблица состоит из строк, определяемыми с помощью тегов **<tr>…</tr>** и внутри строк находятся ячейки, ограниченными тегами **<td>…</td>**.

5. В браузере проверьте как выглядит ваша страница: **http://localhost/mysite** (рисунок 37).

Рис. 37. Внешний вид базовой таблицы в шаблоне страницы

Поначалу внешний вид может не впечатлить, но это признак того, что все сделано правильно. Обратите внимание на название вкладки – правильная кодировка не искажает текст, написанный на русском языке. Если у вас не так, проверьте кодировку страницы (см. п. задания 1.3).

Задание 2. Оформляем шаблон страницы с помощью CSS-стилей.

CSS – текстовый формат описания правил, позволяющий вынести стилевую, оформительскую информацию в отдельный файл. Если вы еще не знакомы с этой технологией, то подробнее можно познакомиться тут – http://htmlbook.ru/samcss. Инструкции стилевого оформления передаются через пары «свойство:значение». Для того чтобы соотнести нужные свойства с определенным тегом в HTML-страницы,

используются классы (**class**). Это специальные идентификаторы, которые разработчик придумывает сам. Название класса может состоять из букв английского алфавита, цифр, нижнего подчеркивания, дефиса. Название класса может начинаться только с буквы английского алфавита.

1. Для того, тобы сделать внешнюю границу таблицы (так сайт зрительно смотрится лучше), можно создать стиль для таблицы (css). Для этого создайте новый файл в текстовом редакторе. Сохраните его в папке **mysite** и назовите **style.css**. В самом файле напишите новый стиль (рис. 38).

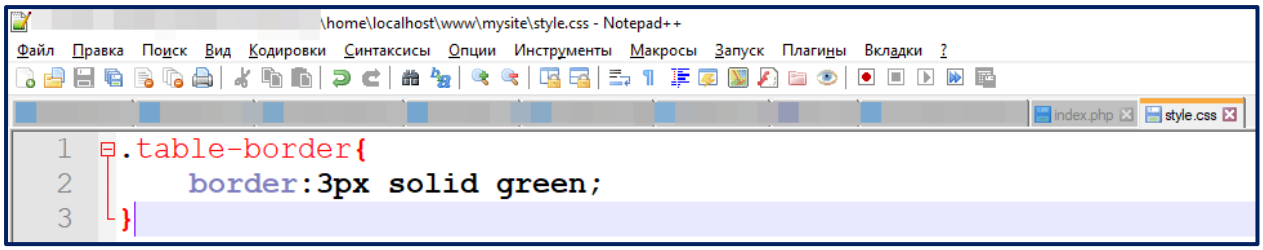


Рис. 38. Стили для таблицы

Пояснение

В этом коде для класса **table-border** задается **border** (граница) толщиной **3 пикселя**, стиль границы **solid** (сплошная) и цвет **green** (зеленый). По желанию, вы можете использовать любой цвет, главное написать его правильно на английском.

2. Теперь нужно подключить данный файл к таблице. Для этого откройте файл **index.php** и добавьте строку (рисунок 39).

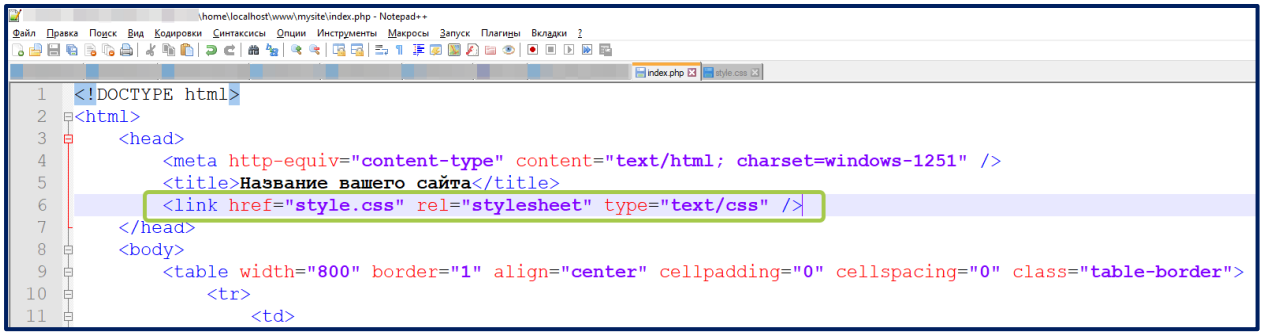


*Рис. 39. Подключение **style.css** к html-файлу*

После этого можно проверить страницу в браузере. В нашем случае должен измениться цвет рамки таблицы (рисунок 40).

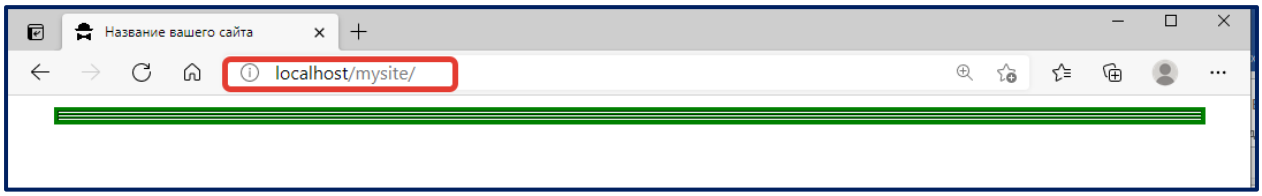


Рис. 40. Стилизация таблицы шаблона

3. Самостоятельно в любом графическом редакторе (Paint, Adobe Photoshop, ArtWeaver, Figma, Photoshop-онлайн) создайте рисунок (или разместите готовый) для «шапки» сайта и самой нижней ячейки таблицы – «подвала» сайта.

Для «шапки» сайта можно использовать размеры: ширина 800 пикселей, высота 250–300 пикселей. В шапке сайта можно разместить логотип, название вашего сайта, ваше имя.

Для подвала сайта можно использовать размеры: ширина 800 пикселей, высота 40–60 пикселей. В подвале сайта можно разместить e-mail разработчика, организации разработчика. Цветовую гамму вы можете выбрать по своему усмотрению. Для примера см. рисунок 41.

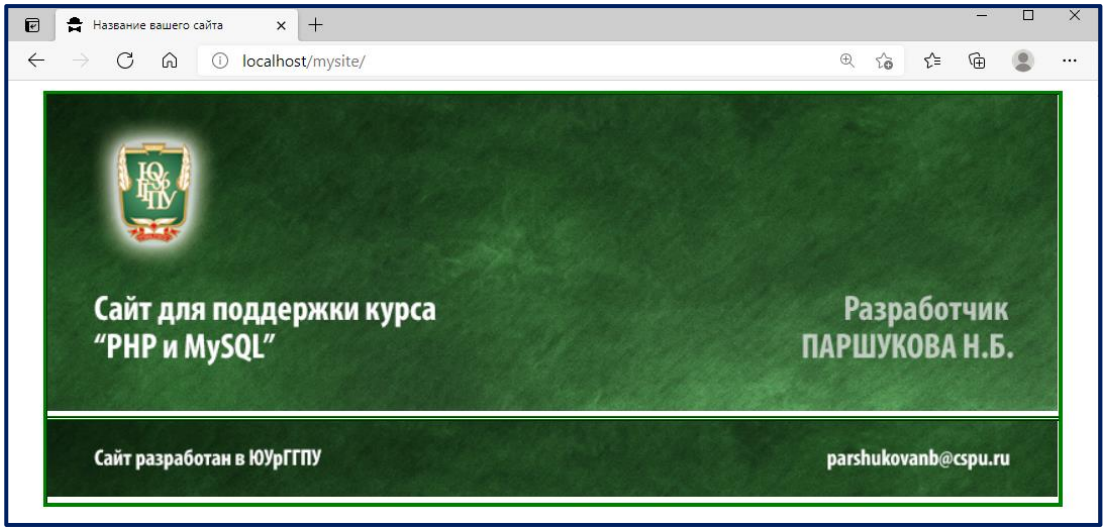


Рис. 41. Внешний вид шаблона главной страницы

Оба изображения нужно разместить в папку **images.** Можно сохранить в формате **jpg, png** или **gif** (рисунок 42).

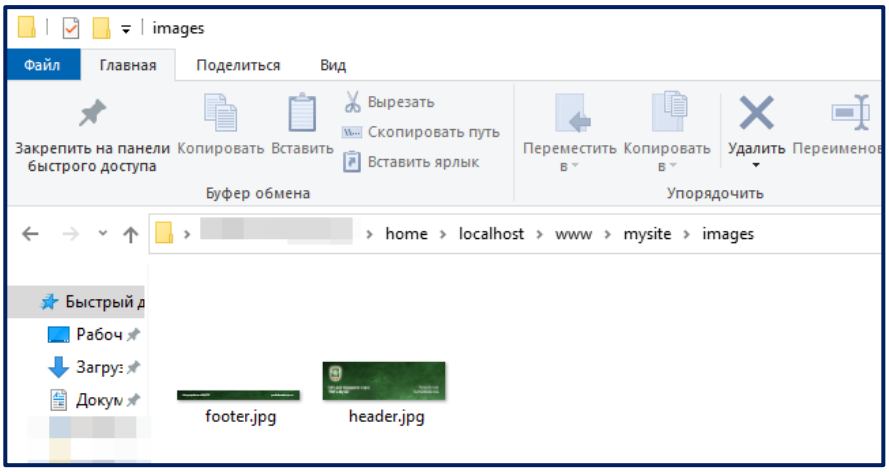


*Рис. 42. Содержимое папки **images***

4. Вставьте рисунок в первую и последнюю ячейку таблицы сайта, где мы указали соответствующие комментарии (рисунок 43).

```
</head>
<body>
    <table width="800" border="1" align="center" cellpadding="0" cellspacing="0" class="table-border">
        <tr>
            <td>
                <!-- В этом месте будет шапка сайта: название сайта и ФИО разработчика -->
                <img src="images/header.jpg" />
            </td>
        </tr>
        <tr>
            <td>

            </td>
        </tr>
        <tr>
            <td>
                <!-- В этом месте будет подвал сайта: информация о правах на копирование, email разработчика -->
                <img src="images/footer.jpg" />
            </td>
        </tr>
    </table>
</body>
```

Рис. 43. Добавление изображений в шапку и подвал сайта

Задание 3. Создаем стили и меню сайта.

1. Все теги будут иметь шрифт Open Sans, размер – 14 пикселей, отступ от всех краев – 15 пикселей. Для этого в файле таблицы стилей **style.css** должна быть прописана следующая строка (рисунок 44).

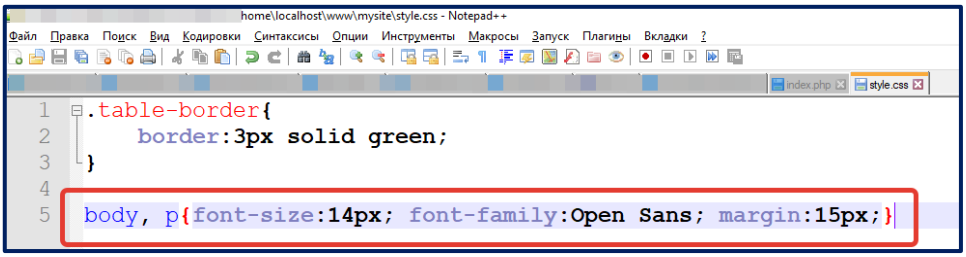


Рис. 44. Стили для абзацев

Проверьте работу со стилями и с PHP: напишите строку с тегами абзаца (** … **) в средней ячейке таблицы (рисунок 45).

```
<table width="800" border="1" align="center" cellpadding="0" cell
    <tr>
        <td>
            <!-- В этом месте будет шапка сайта: название сайта и
            <img src="images/header.jpg" />
        </td>
    </tr>
    <tr>
        <td>
            <?php echo "<p>Протестируем стили</p>" ?>
        </td>
    </tr>
    <tr>
        <td>
            <!-- В этом месте будет подвал сайта: информация о пр
            <img src="images/footer.jpg" />
        </td>
    </tr>
</table>
```

Рис. 45. Тестирование работы php-интерпретатора

Сохраните и посмотрите, что получилось в браузере. Если результат вас устроил, то можно удалить строку с фразой «Протестируем стили», она нужна была только для проверки.

2. В центральную ячейку необходимо вставить еще одну таблицу, которая будет разделять навигационную часть сайта (меню) с контентной (где будет весь текст (статьи). Вставить таблицу, которая будет содержать 1 ряд и 2 колонки. У левой ячейки задать ширину **180** пикселей, а у правой – **620** пикселей. Для выравнивания по вертикали по верху добавим к обеим ячейкам атрибут **valign** со значением **top** (см. рисунок 46).

```
            <!-- В этом месте будет шапка сайта: название сайта и ФИО разработчика -->
            <img src="images/header.jpg" />
        </td>
    </tr>
    <tr>
        <td>
            <table width="800" border="0" cellpadding="0" cellspacing="0">
                <tr>
                    <td width="180" valign="top" class="left-column">
                        <!-- здесь будет меню -->
                    </td>
                    <td width="620" valign="top">
                        <!-- здесь будет контент -->

                        <!-- конец контента -->
                    </td>
                </tr>
            </table>
        </td>
    </tr>
    <tr>
        <td>
            <!-- В этом месте будет подвал сайта: информация о правах на копирование, email разработчика -->
```

Рис. 46. Добавление таблицы в среднюю ячейку таблицы шаблона

3. Задать стилевые свойства для левой колонки: сероватый цвет фона (**background**) и выделенную правую границу (**border-right**). Для этого создать новый стиль, прописав в файле стилей (**style.css**) следующий код (рисунок 47):

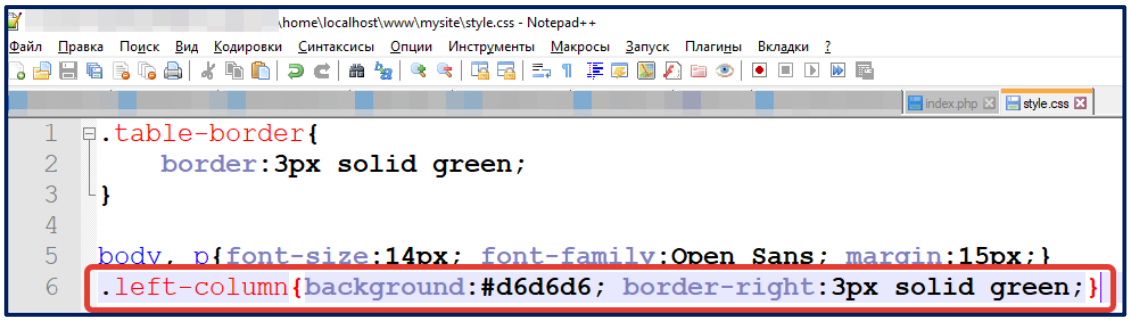


Рис. 47. Стили для левой колонки

4. В отдельном файле создадим меню для всего сайта. Создайте новый файл **menu.php** и сохраните его в папке **mysite**. Проверьте кодировку файла или перекон-вертируйте его через пункт меню Notepad++: **Кодировка / Преобразовать в ANSI** (англ. версия **Encoding / Convert to ANSI**).

Разместите в файле **menu.php** разметку нашего меню:

```
<p class="title">Меню</p>
<ul class="menu">
      <li><a href="index.php">Главная</a></li>
      <li><a href="articles.php">Теория</a></li>
      <li><a href="about.php">Об авторе</a></li>
</ul>
```

5. Чтобы связать информацию из файла **menu.php** и страницу нашего шаблона **index.php,** воспользуемся функцией php **include**.

Пояснение

Функция **include(<filename>)** позволяет включить и выполнить код из файла <filename>. Таким образом можно хранить какую-то информацию или код в одном файле <filename>, а использовать его многократно в других файлах, всего лишь под-ключив конструкцию include(<filename>). В дальнейшем удобно вносить изменения всего лишь в одном файле <filename>. Эти изменения автоматически отобразятся во всех файлах, где используется include(<filename>).

Добавьте указанный на рисунке код в область, где мы планировали разместить меню (рисунок 48).

```
<td>
    <table width="800" border="0" cellpadding="0" cellspacing="0">
        <tr>
            <td width="180" valign="top" class="left-column">
                <!-- здесь будет меню -->
                <?php include("menu.php");?>
            </td>
            <td width="620" valign="top">
                <!-- здесь будет контент -->

                <!-- конец контента -->
            </td>
        </tr>
    </table>
</td>
```

Рис. 48. Подключение меню

6. Для графического оформления меню воспользуемся готовыми таблицами стилей. Перейдите в файл **style.css**. Допишите в него следующий код (цвета можете задать по своему усмотрению).

```
.title{background:green; color:white; padding:5px 15px;}
.menu li{color:green;}
.menu li a{color:green;}
.menu li:hover{color:black}
.menu li:hover a{color:black}
```

Задание 4. Заполняем стартовую страницу контентом (содержимым).

В правой ячейке контентной области (с комментарием **<!--Здесь будет контент -->**) опишите, в чем состоит специфика вашего проекта и чем ваш сайт может быть интересен для потенциальных посетителей.

Вот что примерно должно получиться (рисунок 49).

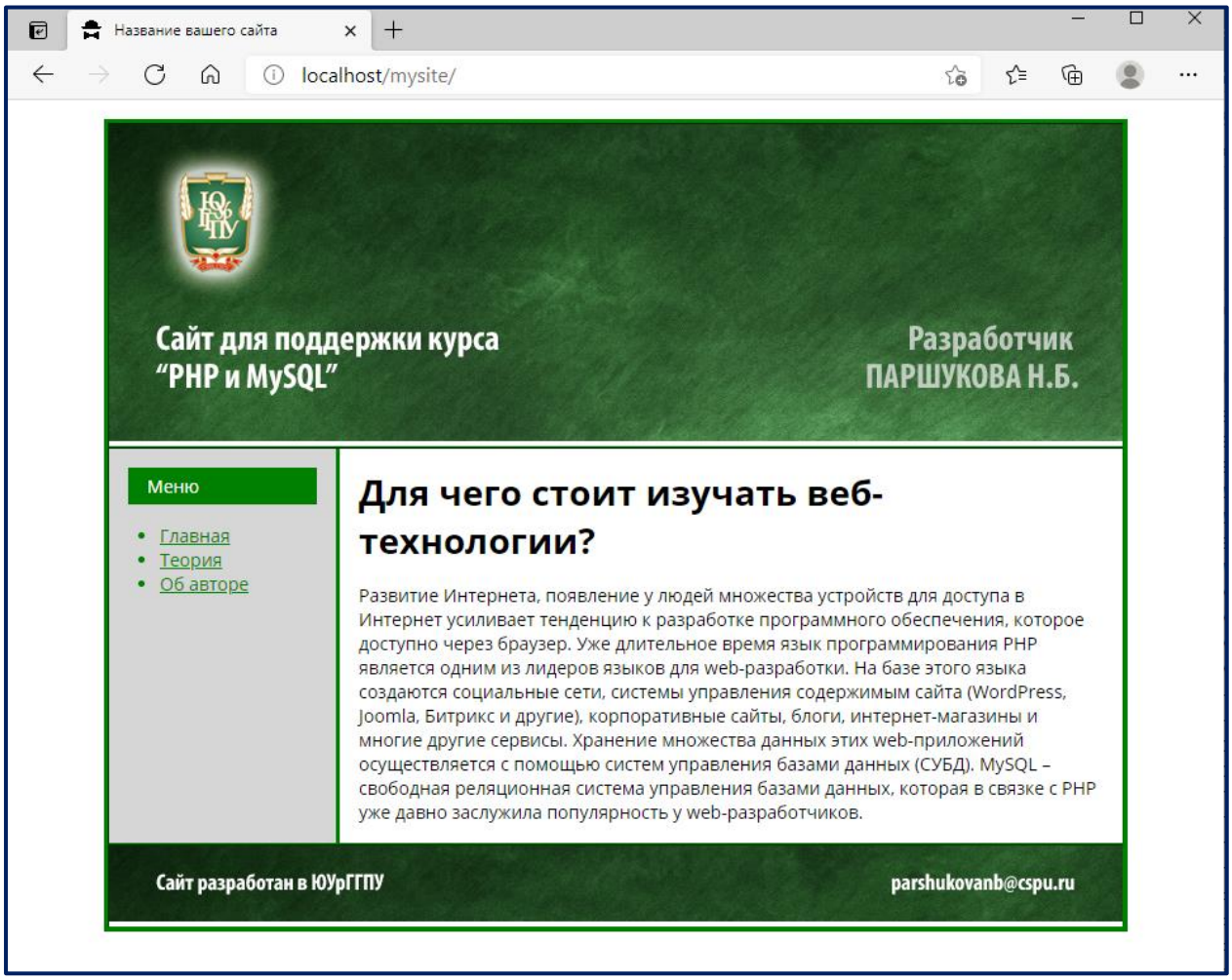


Рис. 49. Внешний вид главной страницы

Задание 5. Сделайте копию файла **index.php** и назовите его **about.php**. Поместите в файл **about.php** текст о вас с изображением (по аналогии с заданием 4, размещайте в контентную область). Проверьте, открывается ли страница **about.php** при переходе в меню.

3.3. Создание базы данных для хранения статей сайта

Задачи лабораторной работы

- Создать БД со статьями по определенной теме;
- Вывести данные из БД в страницу articles.php;
- Создать страницу с шаблоном отдельной статьи article.php для вывода полного содержания урока.

Задание 1. Создадим базу данных для нашего веб-сайта и внесем в одну из ее таблиц информацию о статьях на определенную тему. Вы можете выбрать тематику по своему усмотрению.

1. Запустите phpMyadmin: в строке адреса браузера http://localhost/tools/phpmyadmin.

2. Перейдите на вкладку **Базы данных** и создайте новую базу данных **my_db** (рисунок 50).

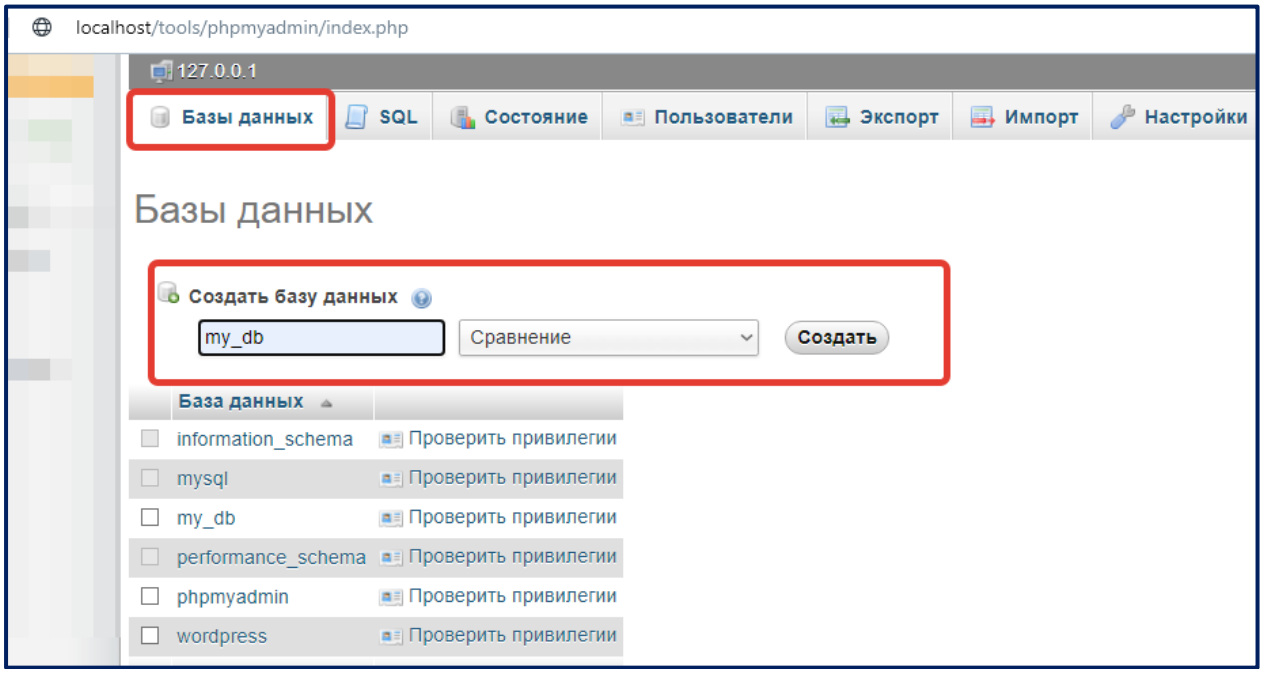


*Рис. 50. Создание базы данных **my_db***

3. В БД **my_db** создайте таблицу **articles** которая будет содержать **8** полей (рисунок 51).

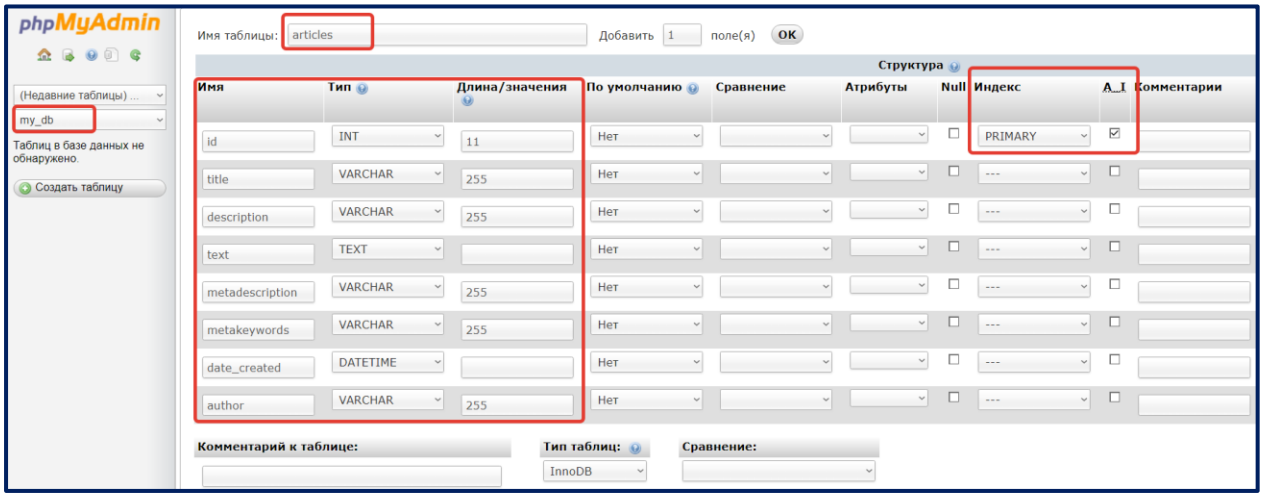


Рис. 51. Создание таблицы articles

Пояснение

id – Порядковый номер статьи (целое 11-значное число – int, первичный индекс – primary, автоматический счетчик – A_I);

title – заголовок статьи (строковый тип данных varchar, длиной 255 символов);

description – краткая (1-2 абзаца) аннотация статьи или ее первый абзац (строковый тип данных varchar, длиной 255 символов, допускается и использование неограниченного типа text);

text – полное содержание статьи, можно использовать теги … для формирования абзацев статьи (неограниченный текстовый тип данных text).

metadescription – мета-описание, необходимое для продвижения через поисковые системы, может совпадать с полем description (строковый тип данных varchar, длиной 255 символов);

metakeywords – необязательное к заполнению поле, но некоторые разработчики его используют для перечисления ключевых словосочетаний для характеристики статьи (строковый тип данных varchar, длиной 255 символов);

date_created – дата создания статьи (тип данных дата-время datetime)

author – автор статьи (строковый тип данных varchar, длиной 255 символов).

4. Сделайте в таблице хотя бы **2 записи**. Для этого выберите пункт меню **Вставить.** Откроется форма для заполнения сразу двух записей. На рисунке показан пример заполнения только одной записи. Т.к. поле **id** мы объявили первичным ключом и автоматическим счетчиком, то его не нужно заполнять – система сделает это сама (рисунок 52).

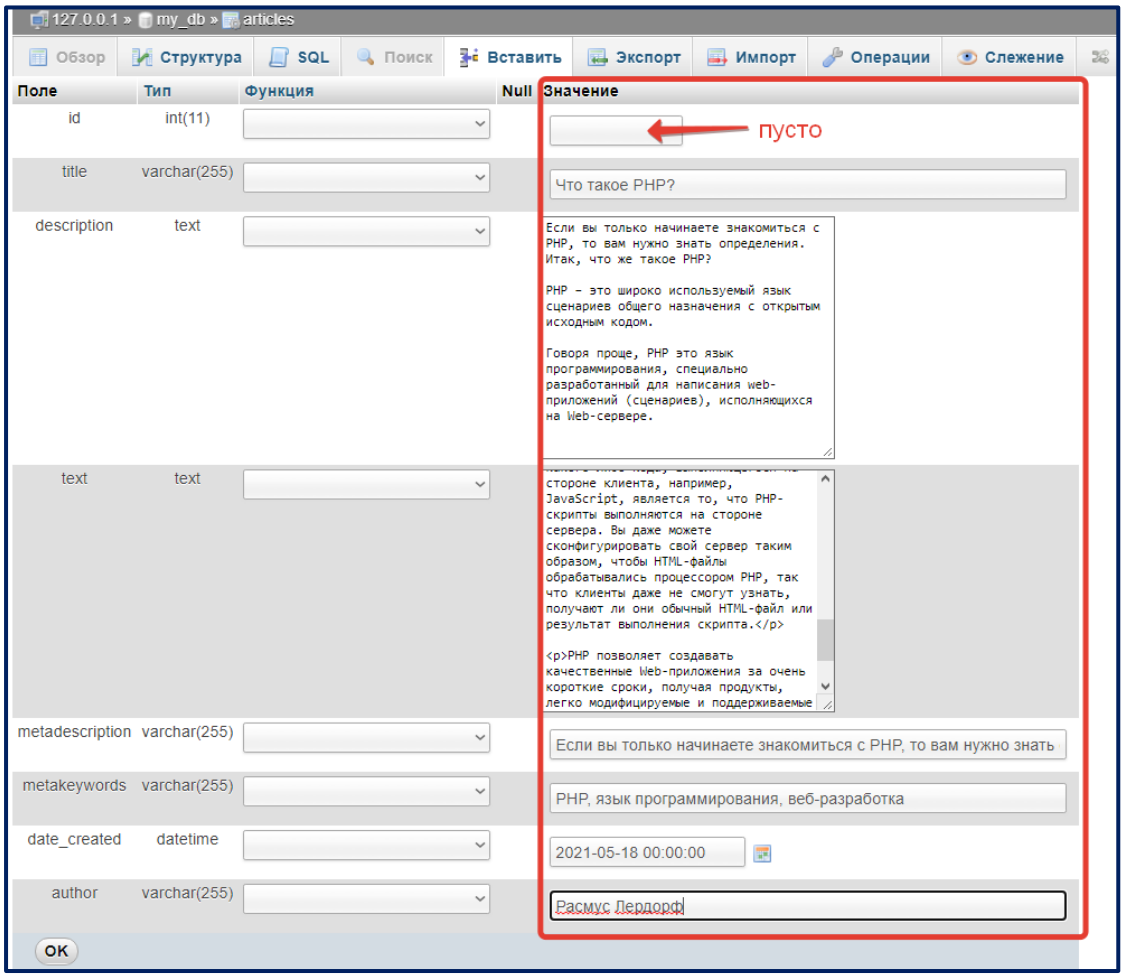


Рис. 52. Вставка записи в таблицу articles

Задание 2. Выведем данные из таблицы articles на страницу articles.php.

1. Сделайте копию файла **index.php** и назовите новый файл **articles.php**. Теперь у вас получились 2 одинаковых файла, но у них разные названия. Удалите содержание контентной части (средняя строка таблицы, второй столбец) – рисунок 53.

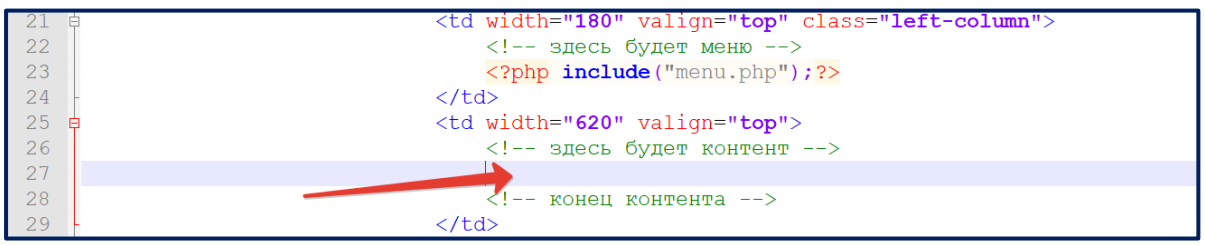


Рис. 53. Добавление комментария в контентной области

2. Создайте файл **db_connect.php**, в котором будет содержаться код соединения с базой данных. В дальнейшей работе нам нужно будет часто осуществлять это соединение, поэтому, чтобы не увеличивать код каждой страницы, мы выносим код соединения с базой данных в отдельный файл. Приведем код этого файла (рисунок 54):

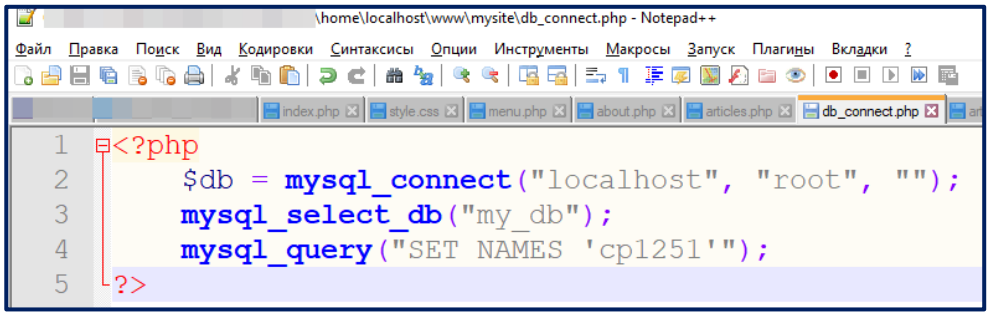


Рис. 54. Содержание файла db_connect.php

Пояснение

mysql_connect – функция, устанавливающая соединение с сервером баз данных. Здесь **localhost** – название сервера, **root** – пользователь базы данных, **«»** – пароль. Конечно, не безопасно использовать пользователя root, у которого нет пароля для управления базой данных. Но при локальной разработке такое допускается.

mysql_select_db – функция, выбирающая на сервере нужную нам базу данных my_db.

mysql_query – функция, выполняющая запрос к базе данных. В данном случае она устанавливает кириллицу cp1251 в качестве предпочтительной кодировки данных.

3. С помощью функции include вставьте код программы **db_connect.php** в самое начало файла **articles.php** (соединение с базой данных должно предшествовать выводу текста веб-страницы, в противном случае соединение может работать некорректно) – рисунок 55.

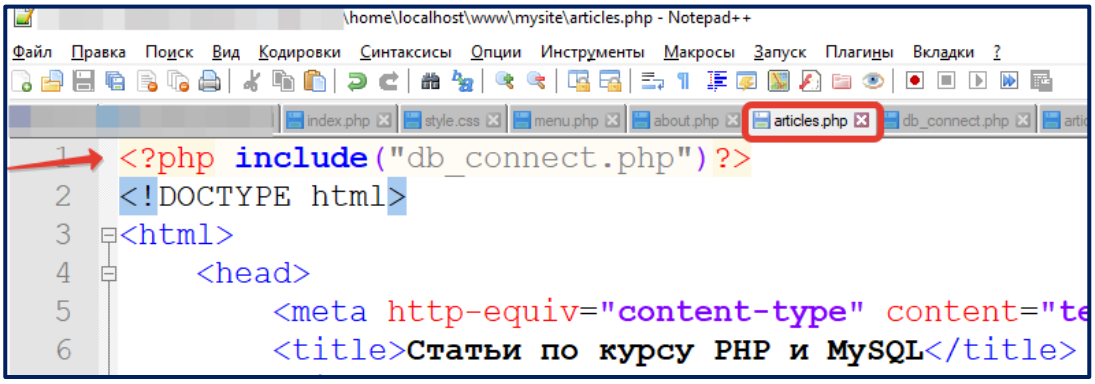


Рис. 55. Добавление подключения файла db_connect.php в файл articles.php

4. В контентной области файла **articles.php** (откуда мы предварительно удалили содержимое) напишем запрос на получение данных из таблицы **articles** (рисунок 56).

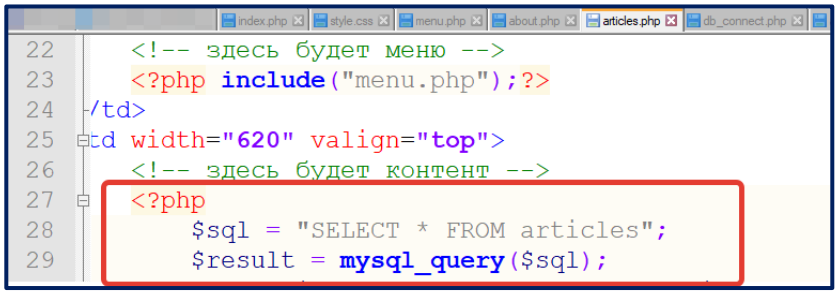


Рис. 56. Формирование запроса к таблице articles

Пояснение

Оператор SELECT позволяет получить данные из таблицы базы данных. Структура его записи в общем виде следующая:

SELECT <поле1>, <поле2>, …, <полеN> FROM <имя_таблицы> WHERE <условие выборки>

В нашем случае вместо перечисления полей мы используем оператор ***,** что означает «выбрать все поля». Выражение WHERE в запросе файла articles.php также отсутствует, что означает «получить все строки из таблицы».

Чтобы запрос начал работать, нужно использовать функцию **mysql_query**, в которую запрос отправляется в качестве параметра.

5. Создадим табличную разметку для вывода краткой характеристики статьи: в левой ячейке выводятся поля заголовок в гиперссылке (title), дата создания статьи (date_created), автор статьи (author); в правой ячейке будет выводиться краткое описание статьи (description). Внешне это будет выглядеть как показано на рисунке 57.

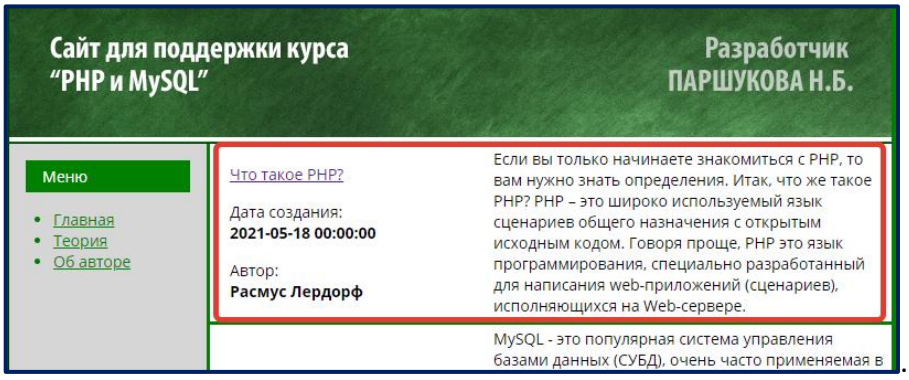


Рис. 57. Шаблон для вывода отдельной статьи на сайте

Программный код таблицы и цикла обработки запроса выглядит следующим образом (рисунок 58):

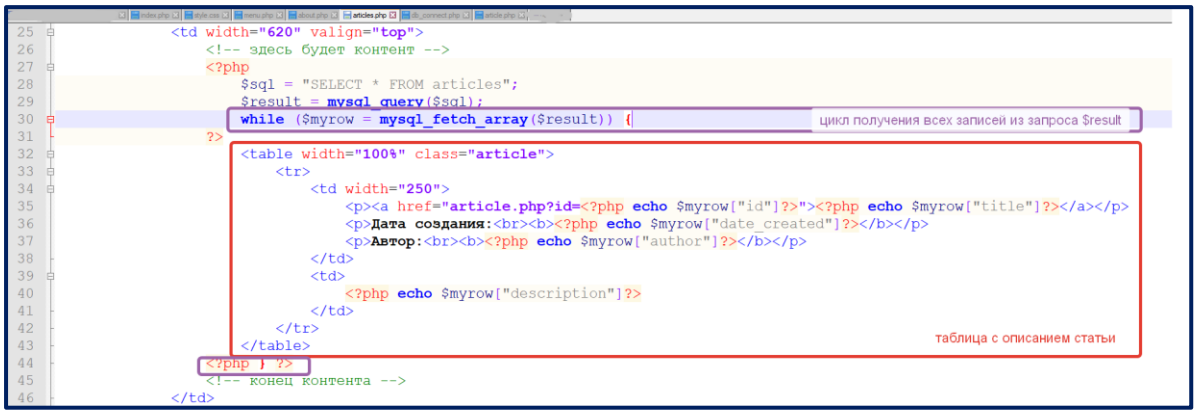


Рис. 58. Цикл выдачи всех статей на сайте

Обратите внимание, что ключи в массиве $myrow должны в точности соответствовать названиям полей вашей таблицы **articles** (см. рисунок 59).

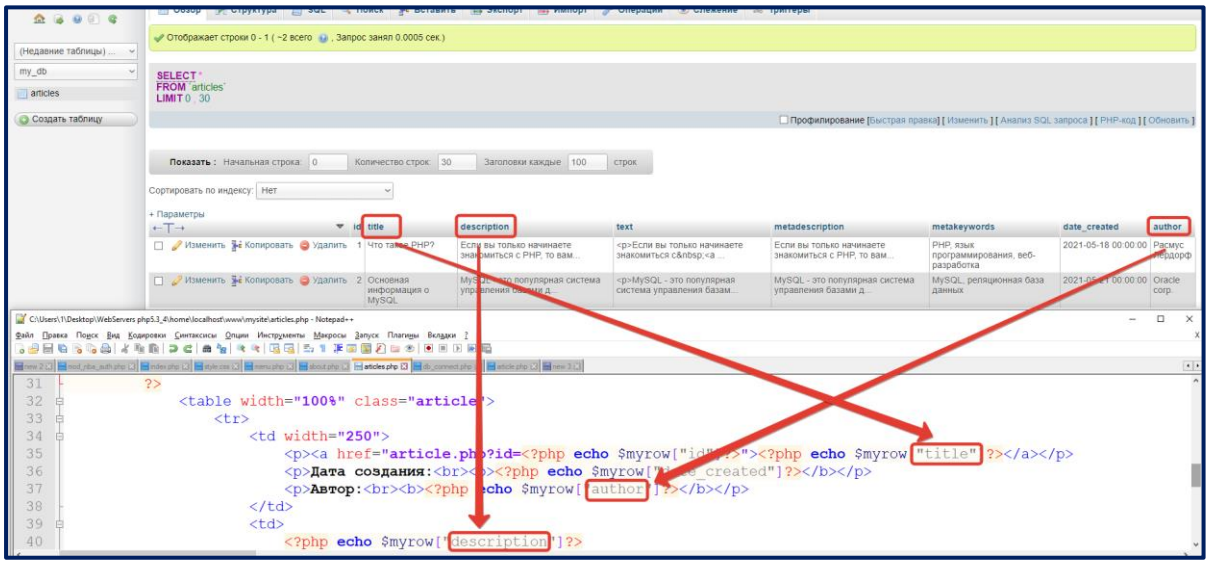
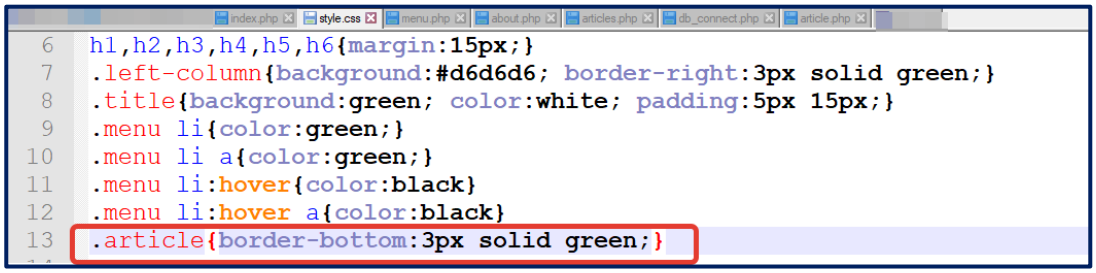


Рис. 59. Связь полей таблицы articles с ключами массива $myrow

6. Для того чтобы таблица выглядела привлекательнее, можно добавить стили (в файл **style.css**) для разделителя таблиц. Цвет рамки вы можете выбрать по своему усмотрению (рисунок 60).



```
h1,h2,h3,h4,h5,h6{margin:15px;}
.left-column{background:#d6d6d6; border-right:3px solid green;}
.title{background:green; color:white; padding:5px 15px;}
.menu li{color:green;}
.menu li a{color:green;}
.menu li:hover{color:black}
.menu li:hover a{color:black}
.article{border-bottom:3px solid green;}
```

Рис. 60. Добавление стилей для шаблона таблицы выдачи отдельной статьи

7. Итоговый результат файла **articles.php** в браузере должен выглядеть примерно так (рисунок 61).

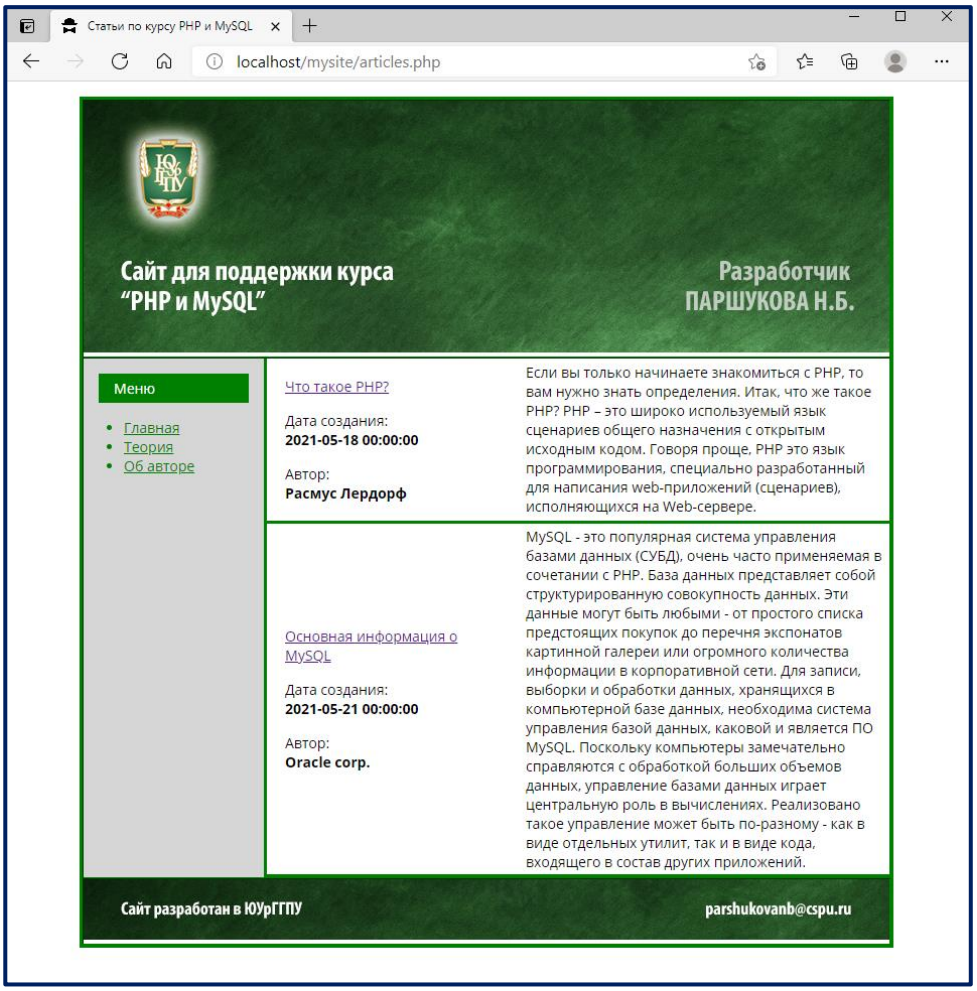


Рис. 61. Внешний вид списка статей файла articles.php

Задание 3. Создадим страницу с шаблоном статьи **article.php** для вывода полного текста статьи.

При выводе таблицы с кратким описанием уроков мы уже создали гиперссылку, с помощью которой в файл **article.php** передается переменная **id**, равная номеру урока в базе, по которому нажали мышкой. Теперь нужно сделать вывод выбранной пользователем статьи полностью.

1. Создайте копию файла **articles.php** и назовите новый файл **article.php** (таким образом, articles.php будет выводить множество статей, а article.php – только одну, что и соответствует их названиям).

Удалите код из контентной части.

2. Сформируйте запрос к базе данных для вывода конкретных полей в зависимости от переданного параметра **id**. Результат всего запроса должен выводиться в переменную, например **$myrow** (рисунок 62).

```php
<?php include("db_connect.php");
    if (isset($_REQUEST["id"])) {
        $id = intval($_REQUEST["id"]);
        $sql = "SELECT * FROM articles WHERE id=$id";
        $result = mysql_query($sql);
        $myrow = mysql_fetch_array($result);
    }
?>
<!DOCTYPE html>
<html>
    <head>
```

Рис. 62. Код получения отдельной записи из таблицы articles

Пояснение

Функция **isset** проверяет, была ли передана в файл **article.php** переменная **id** (значение отличное от NULL). Хотя она передавалась через открытый метод GET (и можно было получить его с помощью массива $_GET), получить ее можно через общий массив **$_REQUEST**, который является более универсальным.

Для краткости значение **$_REQUEST[“id”]** сохраняется в переменную **$id** и, во избежание вредоносных манипуляций, переводится с помощью функции **intval()** в целое число. Для правильного запроса **id** и так должны быть целочисленного значения. Но **intval()** блокирует передаваемые строковые данные, чем нередко пользуются злоумышленники.

В данном коде запрос получает из таблицы **articles** единственную запись, **id** которой соответствует передаваемому значению **id**. Если передано значение 1, то отобразится статья, имеющая в таблице id = 1.

3. В результате запроса мы получили ассоциативный массив $myrow, в качестве ключей которого используются названия полей из таблицы **articles**. Выведем эти значения в соответствующие теги нашего файла **article.php**.

Для начала выведем заголовок нашей статьи в тег <title> … </title> (рисунок 63).

```php
        <meta http-equiv="content-type" content="text/html; charset=windows-1251" />
        <title><?php echo $myrow["title"]?></title>
        <link href="style.css" rel="stylesheet" type="text/css" />
```

Рис. 63. Вывод заголовка в теге <title>…</title>

Метатеги (metadescription и metakeywords) разместим в специальных тегах внутри раздела head файла **article.php** (рисунок 64):

```php
    <head>
        <meta name="description" content="<?php echo $myrow["metadescription"]?>">
        <meta name="keywords" content="<?php echo $myrow["metadescription"]?>">
        <meta http-equiv="content-type" content="text/html; charset=windows-1251" />
        <title><?php echo $myrow["title"]?></title>
        <link href="style.css" rel="stylesheet" type="text/css" />
    </head>
```

Рис. 64. Вывод мета тегов на странице

Внутри контентной области разместите вывод заголовка статьи в теге <h1>…</h1>, дата создания, автора и полный текст статьи (рисунок 65).

```
31                  <td width="180" valign="top" class="left-column">
32                      <!-- здесь будет меню -->
33                      <?php include("menu.php");?>
34                  </td>
35                  <td width="620" valign="top">
36                      <!-- здесь будет контент -->
37                      <h1><?php echo $myrow["title"]?></h1>
38                      <p>Дата создания:<br><b><?php echo $myrow["date_created"]?></b></p>
39                      <p>Автор:<br><b><?php echo $myrow["author"]?></b></p>
40                      <?php echo $myrow["text"]?>
41                      <!-- конец контента -->
42                  </td>
```

Рис. 65. Вывод контента на странице с отдельной статьей

4. Проверьте работу всего сайта. Отдельная страница со статьей должна выглядеть примерно так (рисунок 66).

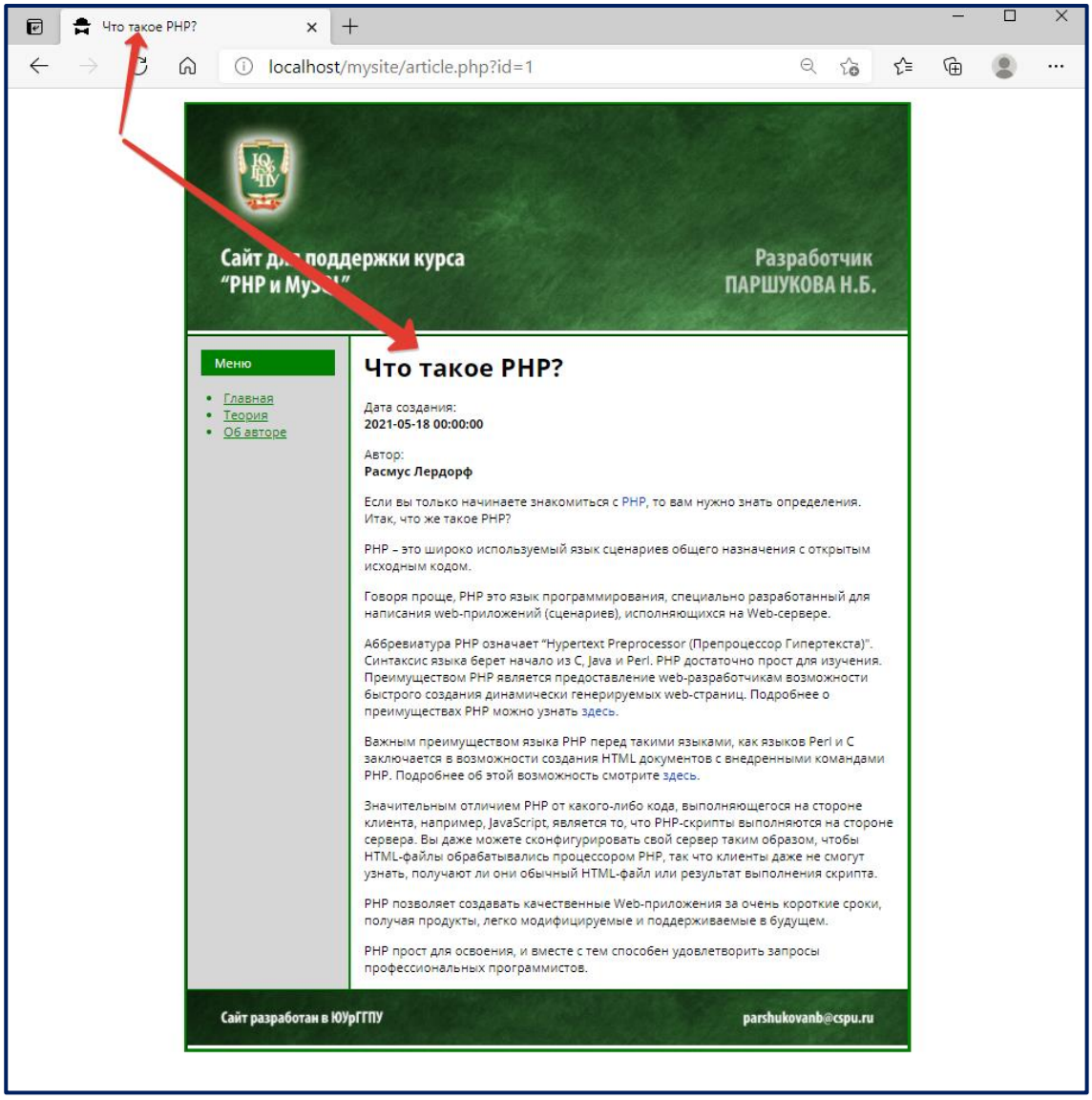


Рис. 66. Вывод отдельной статьи

3.4. Создание гостевой книги для сайта

Вопросы для повторения

1. Что такое HTML-формы?
2. Какие методы передачи данных от браузера на сервер существуют?
3. Как осуществить передачу данных с HTML-формы php-скрипту?
4. Как записать запрос на выборку данных с учетом сортировки по определенному полю?
5. Как сформулировать запрос на вставку новой записи в таблицу?
6. Как сформулировать запрос на удаление записи из таблицы?

Пояснение

Ранее мы уже познакомились с запросом на выборку SELECT, с помощью которого получили полный список статей из таблицы, а также отдельную статью. Рассмотрим в общем виде и примеры четырех основных запросов к таблицам в базе данных: **SELECT**, **INSERT**, **UPDATE** и **DELETE**.

Общи вид запроса на выборку следующий:

```
SELECT <что_выбираем>
FROM <имя_таблицы>
[WHERE <условие_для_выборки>]
[ORDER BY <название_поля> [ASC | DESC]]
[LIMIT [offset,] rows]
]
```

<что_выбираем> может быть перечнем полей, либо *, которая заменяет выбор всех полей.

Сравните

```
SELECT id, title, description, text, author FROM articles
```

Или

```
SELECT * FROM articles
```

Выражения **WHERE**, **ORDER BY** и **LIMIT** не являются обязательными в разделе. Именно поэтому в нашем запросе они заключены в квадратные скобки.

Выражение **WHERE** позволяет задать условие выборки. Если нужно составить сложное условие, то могут быть использованы операторы **AND** (логическое «И»), **OR** (логическое «ИЛИ») и **NOT** (логическое «НЕ»).

Пример, запрос на выборку всех записей со значением поля **id** равное 2 и поля **author** содержащего 'Расмус Лердорф':

```
SELECT * FROM articles WHERE id = 2 AND author='Расмус Лердорф'
```

Выражение **ORDER BY** используется для сортировки записей в вашем результирующем наборе. Здесь обязательно указывается поле для сортировки и тип сортировки: по возрастанию (**ASC**) и по убыванию (**DESC**). Если атрибуты **ASC** или **DESC** не указаны в операторе **ORDER BY**, результаты будут отсортированы по полю в порядке возрастания. Это эквивалентно выражению **ORDER BY <поле> ASC**.

Например, следующий запрос отсортирует сначала в порядке убывания по автору, затем – по названию в порядке возрастания:

```
SELECT * FROM articles ORDER BY author DESC, title
```

Выражение **LIMIT** может использоваться для ограничения количества строк, возвращенных командой **SELECT**. **LIMIT** принимает один или два числовых аргумента. Эти аргументы должны быть целочисленными константами. Если заданы два аргумента, то первый указывает на начало первой возвращаемой строки, а второй задает максимальное количество возвращаемых строк. При этом смещение начальной строки равно 0 (не 1).

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

возвращает строки 6–15.

Запрос на вставку **INSERT** позволяет вставить новые записи в таблицу. Общий вид запроса:

```
INSERT INTO <имя_таблицы>
(<поле1>, <поле2>, ... )
VALUES (<выражение1>, <выражение2>, ... )
```

Например,

```
INSERT INTO articles (id, title) VALUES (3, 'Синтаксис запроса INSERT')
```

вставит в таблицу **articles** строку с **id** равным **3** и заголовком «Синтаксис запроса INSERT». Остальные поля будут заданы значениями по умолчанию (чаще всего это NULL, 0 или пустая строка).

Запросы на обновление имеющихся записей **UPDATE**. Его синтаксис следующий

```
UPDATE <имя_таблицы>
SET <поле1> = <выражение1>, <поле2> = <выражение2>,
... [WHERE <условие_для_выборки>]
```

Здесь, выражение **WHERE** не является обязательным, но если не задуматься над условием для выборки, то обновятся все записи в таблице.

Например, запрос

```
UPDATE articles SET date_created='2021-15-05 00:00:00', author='Oracle corporation' WHERE id = 1
```

обновит дату создания и имя автора для записи с **id = 1**. Не указанные в запросе поля останутся без изменения.

Запрос на удаление записей **DELETE** в общем виде выглядит так

```
DELETE FROM <имя_таблицы> [WHERE <условие_для_выборки>]
```

Например, удалит записи о статьях, созданных позже 15 мая 2021 года, запрос

```
DELETE FROM articles WHERE date_created > '2021-05-15'
```

Пояснение

Гостевая книга – это специальный раздел или страница на сайте, где посетители могут оставить свои сообщения на вашем ресурсе.

Логика работы гостевой книги очень простая – выводится список сообщений, предпочтительный порядок сортировки от новых сообщений к более старым. Каждый посетитель через специальную форму может отправить новое сообщение. Также для нашего проекта мы реализуем возможность удаления отдельного сообщения.

Задание 1. Создадим таблицу **guest_book** в базе данных **my_db** для хранения сообщений гостевой книги.

1. Запустите phpMyadmin: http://localhost/tools/phpmyadmin.
2. На рисунке 34 показан процесс создания таблицы **guest_book**. При ее создании указывайте ТОЛЬКО **имя поля**, **тип поля** и **длину поля**. Для ключевого поля (**id**) добавьте в столбце **Индекс** значение **PRIMARY** (первичный ключ), поставьте галочку в столбце **A_I** (auto_increment, счетчик) – рисунок 67.

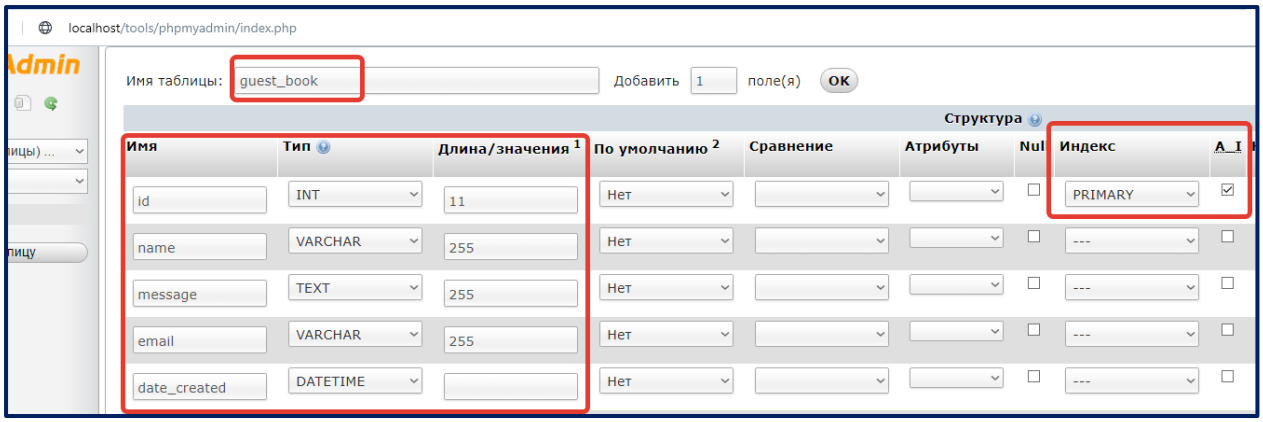


Рис. 67. Создание таблицы guest_book

3. Сделайте в таблице не менее 2-х записей тестовых сообщений. Поле **id** не нужно заполнять, т.к. уникальные записи первичного ключа таблицы система контролирует сама (рисунок 68).

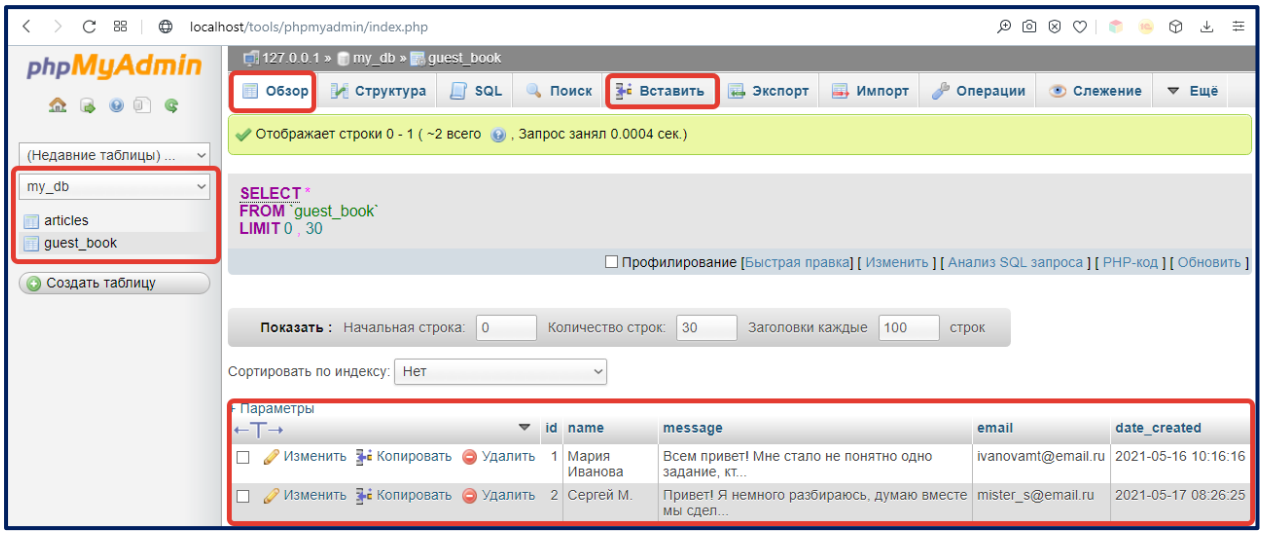


Рис. 68. Пример заполнения таблицы guest_book

4. Добавьте новый пункт меню в файл **menu.php** «Гостевая книга». Укажите ссылку на файл **guest_book.php**.

5. Создайте файл **guest_book.php** в папке с вашим сайтом (на основе файла для вывода статей **articles.php**).

6. Внешний вид файла **guest_book.php** должен выглядеть следующим образом (рисунок 36). Установите нужные для гостевой книги значения тегов **<title>…</title>** и заголовка **<h1>…</h1>** контентной части (рисунок 69).

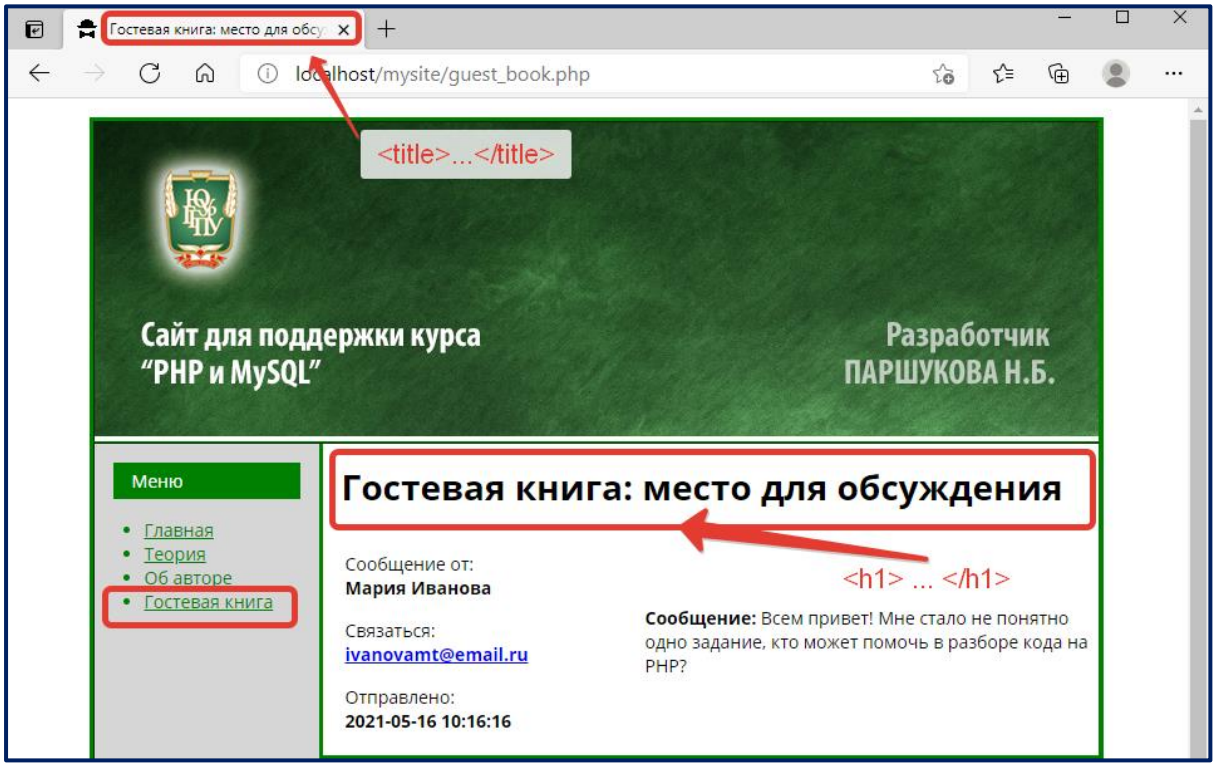


Рис. 69. Внешний вид гостевой книги

7. Сформулируйте запрос на выборку всех полей таблицы **guest_book** с сортировкой по полю **date_created**. Вид сортировки по убыванию – **DESC**.

8. Измените тело цикла выдачи значений из базы данных, чтобы оно соответствовало рисунку. Т.к. алгоритм очень похож на выдачу информации о статьях, вам нужно вывести в нужные строки поля **$myrow['name'], $myrow['email'], $myrow['message'], $myrow['date_created']**.

9. В скрипте после завершения цикла выдачи создайте форму отправки сообщения, состоящую из полей: имя, e-mail (текстовые поля), сообщение (textarea), кнопка «Отправить» (submit).

```
<form action="guest_book_handler.php?task=send" method="post">
      <p>Ваше имя<br/><input type="text" name="name" /></p>
      <p>Ваш email<br/><input type="text" name="email" /></p>
      <p>Ваше сообщение<br/><textarea name="message" rows="5" cols="30"
/></textarea></p>
      <p><input type="submit" name="button" value="Отправить"></p>
</form>
```

Обратите внимание, что обработчиком формы отправки сообщений является скрипт **guest_book_handler.php** (от слова *handler* – обработчик), который нам еще предстоит создать. После знака «?» находится переменная **task** («задача») со значением **send** («отправить»). По этой переменной скрипт **guest_book_handler.php** будет определять, какую задачу ему нужно будет выполнять: заносить ли новое сообщение в таблицу базы данных, удалять сообщение или что-то другое.

10. Проверьте работу скрипта **guest_book.php**. Должна появиться форма после всех сообщений гостевой книги.

11. Создайте новый файл и назовите его **guest_book_handler.php**. Проверьте работу добавления сообщения (рисунок 70).

```php
<?php
    include("db_connect.php");
    if (isset($_GET["task"])) $task = $_GET["task"];
    switch ($task) {
        default:
        case "send":
            sendMessage();
            break;
        case "delete":
            deleteMessage();
            break;
    }

    function sendMessage(){
        $name       = addslashes($_POST["name"]);
        $email      = addslashes($_POST["email"]);
        $message    = addslashes($_POST["message"]);
        $result     = mysql_query("INSERT INTO guest_book (name,email,message,date_created)
                                   VALUES ('$name','$email','$message',NOW())");
        if ($result) {
            header("location:guest_book.php");
        }
    }

?>
```

Рис. 70. Код отправки сообщения в гостевую книгу

Пояснение

В приведенном алгоритме используется функция **addslashes**, которая экранирует строку с помощью слешей, позволяет заменить все внутренние строковые кавычки и апострофы и обратные слеши, добавив к ним символ \. Это позволяет избежать конфликтов при вставке записей в базу данных строк, содержащих спецсимволы кавычек, апострофов, обратных слешей и др.

Также сетевая функция **header("location:<адрес>")** выполняет принудительный редирект (перенаправление) на нужную нам страницу, чтобы пользователь без лишних движений был перенаправлен на нужную страницу.

12. Для удаления отдельного сообщения сделаем гиперссылку с небольшим рисунком крестика, по нажатию на который будет происходить удаление выбранного сообщения. Найдите в сети Интернет подходящее изображение и сохраните его в папке **images** (в которую ранее мы уже сохраняли изображения для нашего сайта). Название файла с крестиком сделайте максимально простым, в нашем примере файл называется **close.png** (рисунок 71).

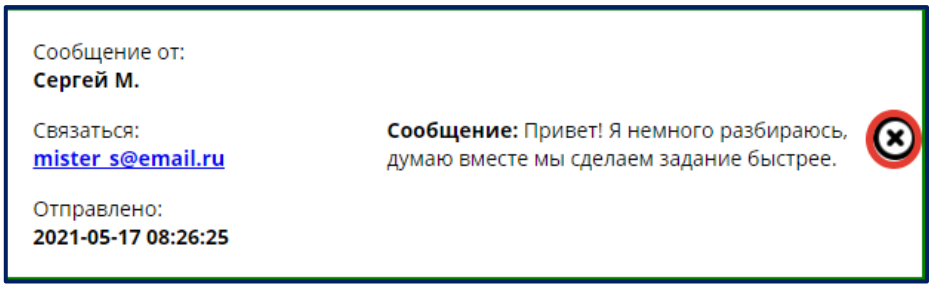


Рис. 71. Пиктограмма для удаления записи гостевой книги

Для этого в файле **guest_book.php** в цикле вывода таблицы с отдельным сообщением гостевой книги вставьте следующий код html:

```
<a href="guest_book_handler.php?task=delete&id=<?php echo $my-
row["id"]?>"><img src="images/close.png" alt="Удалить сообщение"
width="30" align="right"></a>
```

Обратите внимание на гиперссылку (тег **<a href>**). В теге <a> (гиперссылке) есть возможность переходить на другую страницу. Для этого используется атрибут **href**. Для удаления сообщения нужно передать скрипту обработчику **guest_book_handler.php** два параметра – **task** со значением **delete** («удалить») и **id** записи сообщения, которое мы будем удалять.

13. Самостоятельно допишите в файле **guest_book_handler.php** функцию для удаления выбранной записи. Ее можно сделать по аналогии с функцией **sendMessage()**, только для удаления нам нужно знать только **id** выбранной записи.

14. Проверьте работу гостевой книги и покажите результат преподавателю.

3.5. Авторизация пользователя на сайте

Вопросы для повторения

1. Какие существуют типы SQL-запросов?
2. Какие методы передачи данных от клиента (браузера) на сервер существуют?
3. Что понимается под сессией в PHP?
4. Как начать работать с сессией?
5. Какой массив используется для хранения переменных сессий?

Задачи лабораторной работы

- Создать таблицу базы данных с информацией о зарегистрированных пользователях;
- Создать форму для авторизации;
- Разработать алгоритмы для авторизации пользователя и выхода из режима авторизованного пользователя.

Ход работы

Задание 1. В базе данных **my_db** создайте таблицу **users** с полями **id** (целое, счетчик, ключевое), **login** (20 символов), **password** (20 символов). Эту таблицу заполнять не нужно.

Задание 2. На всех страницах нашего сайта разместим форму для авторизации. Для этого:

1. Достаточно добавить в файл **menu.php** подключение к файлу **auth.php**

```
<?php include("auth.php")?>
```

2. В конечном итоге форма авторизации будет отображаться на страницах нашего сайта в двух состояниях:

- если пользователь не вводил логин и пароль, то есть еще не был авторизован, то ему предлагается форма для ввода логина и пароля (рисунок 72);

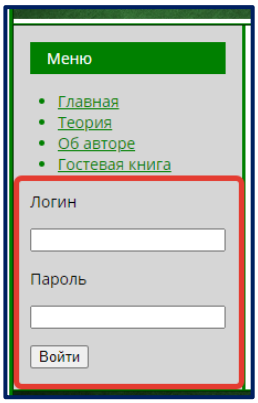


Рис. 72. Форма для ввода логина и пароля

- если же пользователь один раз ввел логин и пароль, то алгоритм должен это запомнить и приветствовать пользователя, а также предлагать пользователю возможность сбросить авторизацию (рисунок 73).

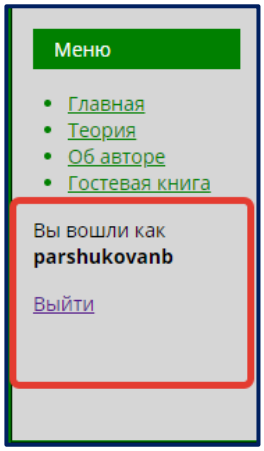


Рис. 73. Состояние блока при авторизованном пользователе

Пояснение

Вопрос в нашем простом алгоритме авторизации заключается в механизме «запоминания» авторизованного пользователя. Дело в том, что протокол HTTP – это протокол без учета состояния, что означает, что сервер не может запоминать конкретного пользователя между несколькими запросами. Например, при доступе к веб-странице сервер отвечает за предоставление содержимого запрашиваемой страницы. Поэтому, когда вы обращаетесь к другим страницам одного и того же веб-сайта, веб-сервер интерпретирует каждый запрос отдельно, как если бы они не были связаны друг с другом. Серверу не известно, что каждый запрос исходит от одного и того же пользователя. В результате пользователь, авторизовавшись на одной странице, будет вынужден снова вводить логин и пароль для другой страницы. А это, мягко говоря, не удобно.

Сессия позволяет обмениваться информацией на разных страницах одного сайта или приложения, что помогает поддерживать состояние. Это позволяет серверу знать, что все запросы исходят от одного и того же пользователя, что дает возможность отображать информацию и настройки пользователя. Сессии – это механизм, позволяющий однозначно идентифицировать **браузер** и создающий для этого браузера **файл** на сервере, в котором хранятся переменные сеанса. Файлы сессий хранятся в папке **tmp** вашего веб-сервера.

Чтобы начать работать с переменными сессии, нужно убедиться, что сессия запущена. Сделать это можно с помощью функции **session_start()**, которую вызывают в самом начале файла (до вывода HTML-кода страницы). Иногда это удобно делать в файле, где происходит подключение к базе данных, ведь его мы тоже подключаем к каждой странице нашего сайта через <?php include("db_connect.php")?>.

После запуска сессии инициализируется суперглобальный массив **$_SESSION** с помощью соответствующей информации о сессии. По умолчанию он инициализируется пустым, и вы можете хранить дополнительную информацию с помощью пары ключ – значение. Например,

```
$_SESSION["login"] = 'ivanovma'; //инициализация переменной сессии
echo $_SESSION["login"];         //вывод значения переменной сессии
```

3. Создайте файл **auth.php**, который будет осуществлять авторизацию пользователя (не забудьте задать правильную кодировку страницы в Notepad++ **преобразовать в ANSI**) – рисунок 74.

```
<?php
    if (isset($_POST["login"]) && isset($_POST["password"])){
        $login      = $_POST["login"];
        $password   = $_POST["password"];
        $sql = "SELECT * FROM users WHERE login = '$login' AND password = '$password'";
        $result = mysql_query($sql);
        if (mysql_num_rows($result)==0){
            $sql = "INSERT INTO users (login, password) VALUES ('$login', '$password')";
            $result = mysql_query($sql);
        }
        $_SESSION["login"] = $login;
    } else {
        if (isset($_SESSION["login"])){
            $login = $_SESSION["login"];
        }
    }
    if (isset($login)) { ?>
        <p>Вы вошли как <b><?php echo $login;?></b></p>
        <p><a href="logout.php">Выйти</a></p>
    <?php } else { ?>
        <form method="post">
            <p>Логин</p>
            <p><input type="text" name="login"></p>
            <p>Пароль</p>
            <p><input type="password" name="password"></p>
            <p><input type="submit" value="Войти"></p>
        </form>
    <?php } ?>
```

Рис. 74. Код авторизации

Пояснение

Алгоритм работы скрипта, идентифицирующего пользователя, будет следующим:

- Если из формы методом **POST** были переданы значения переменных login и password, то:
  - проверяем, имеется ли пользователь **login** с паролем **password** в таблице **users**;
    - если пользователь отсутствует в таблице **users**, то заносим информацию о логине и пароле пользователя в таблицу **users**;
  - создаем сессию и записываем туда информацию о пользователе – достаточно в сессии хранить только **login**, это позволяет минимизировать риски утечек паролей;
- Если из формы методом POST не были получены **login**, **password**, то:
  - нужно посмотреть, существует в сессии значения $_SESSION["login"];
- Переменная **$login** может получить значение авторизованного пользователя либо из данных формы авторизации, либо из сессии. Если **$login** имеет значение, то:
  - приветствуем зарегистрированного пользователя;
- иначе (**$login** не получила никакого значения) выдаем форму авторизации пользователя.

4. Добавьте функцию **session_start()** в файл **db_connect.php** – рисунок 75.

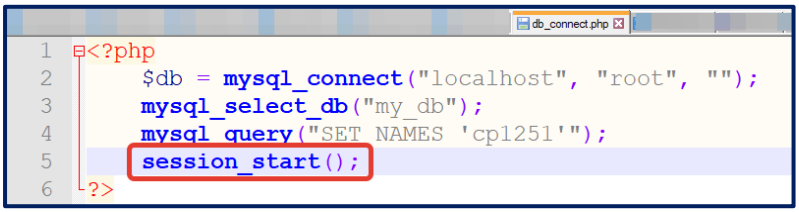


Рис. 75. Код файла db_connect.php

Также проверьте, чтобы запись

```
<?php include("db_connect.php")?>
```

была добавлена в самое начало файлов страниц сайта **index.php**, **about.php** и других файлов вашего сайта, если вы их создавали дополнительно.

5. Проверьте работу авторизации. Если в процессе выполнения алгоритма возникли ошибки, то внимательно посмотрите, с чем они связаны. Еще раз проверьте наличие вызова **session_start()**. Проверьте, какие запросы отправляются в таблицу **users** (очень частая ошибка – несоответствие названий полей в таблицах базы данных и названий полей в запросах).

6. Проверьте, что в момент авторизации и сохранения информации в сессии на вашем локальном сервере сохраняется файл с сессией. Найти его можно в папке **tmp** вашего сервера (рисунок 76), все файлы при этом можно отсортировать по дате по убыванию.

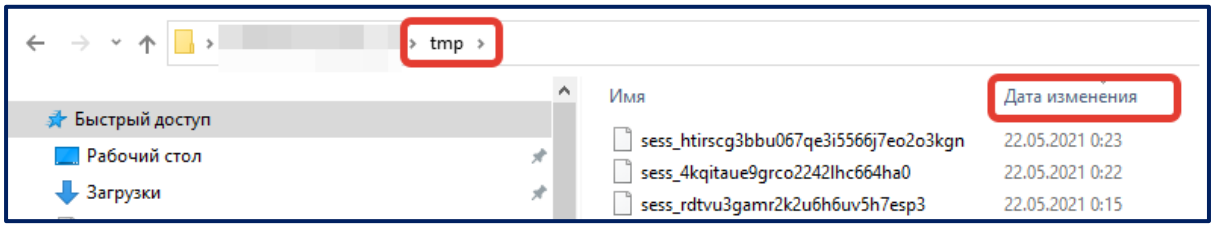


*Рис. 76. Хранение сессий в папке **tmp***

Задание 3. При нажатии на гиперссылку «Выйти» должна происходить очистка сессии от данных авторизованного пользователя.

1. Для этого нужно создать скрипт **logout.php** со следующим кодом (рисунок 77):

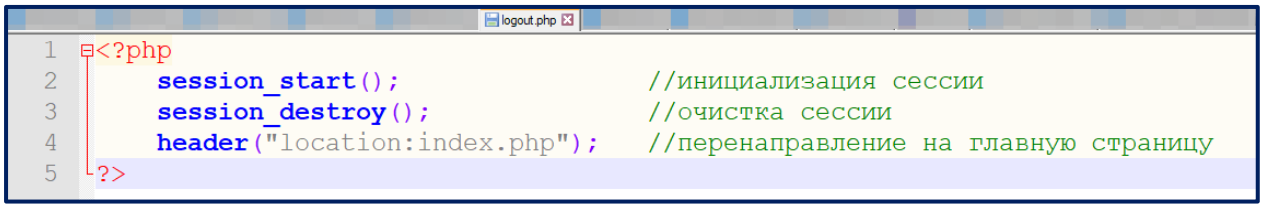


Рис. 77. Код файла logout.php

2. Проверьте работу авторизации: вход пользователя и выход. Проверьте, что авторизация не сбрасывается при переходе на другие страницы сайта.

Задание 4. Улучшение алгоритма авторизации.

В приведенном алгоритме авторизации есть существенный недостаток: пароли пользователей хранятся в открытом виде, и владелец сайта всегда может увидеть, какие пароли пользователи придумывают. Обычно пароль в базе данных хранится в зашифрованном виде. Самым простым вариантом шифрования пароля является алгоритм **md5().** Пример использования функции md5():

```
<?php
$str = '123456';
if (md5($str) === 'e10adc3949ba59abbe56e057f20f883e') {
    echo "строка и хэш совпали";
}
?>
```

Разработайте алгоритм, в котором хранение паролей в базе данных осуществляется в зашифрованном виде (рисунок 78).

id	login	password
3	ivanovma	e10adc3949ba59abbe56e057f20f883e
4	parshukovanb	1bbd886460827015e5d605ed44252251
5	petrovsd	1bbd886460827015e5d605ed44252251

Рис. 78. Пример хранения хешей вместо паролей

А проверка пароля **$password**, который ввел пользователь через форму, происходит через хеширование **$password** и сравнение с хешем из базы данных. При этом, если для определенного пользователя пароль был введен неправильно, то система должна выдавать сообщение, что пароль неверен и предлагать ввести логин и пароль снова (вам может понадобиться еще одна сессионная переменная, например, **$_SESSION["auth"]**, которая может хранить два значения: **1** – если пароль был задан правильно и **2** – если пароль был задан неправильно).

3.6. Создание контрольных вопросов для сайта

Вопросы для повторения

1. Какие существуют типы SQL-запросов?
2. В чем отличия методов передачи данных GET и POST?
3. Как вывести элемент формы «радиоточку» (radio button)?
4. Как можно передать значения радиоточки с формы?
5. Как вывести на форме скрытое поле? Для каких задач предназначено скрытое поле?

Задачи лабораторной работы

- Создать таблицу базы данных с содержанием вопросов теста для каждой статьи.
- Вывести данные из таблицы базы данных на страницу.
- Обработать результаты тестирования и вывести статистику по данному пользователю.

Пояснение

В этой лабораторной работе мы создадим небольшой тест, который предназначен для проверки знаний по отдельной статье нашего сайта. Для каждой статьи будут выводиться свои вопросы. После ответов на вопросы пользователь отправит форму, и в результате будут выведены результаты тестирования. Каждый тестовый вопрос содержит 2–6 вариантов ответа и предполагает выбор только одного варианта из нескольких.

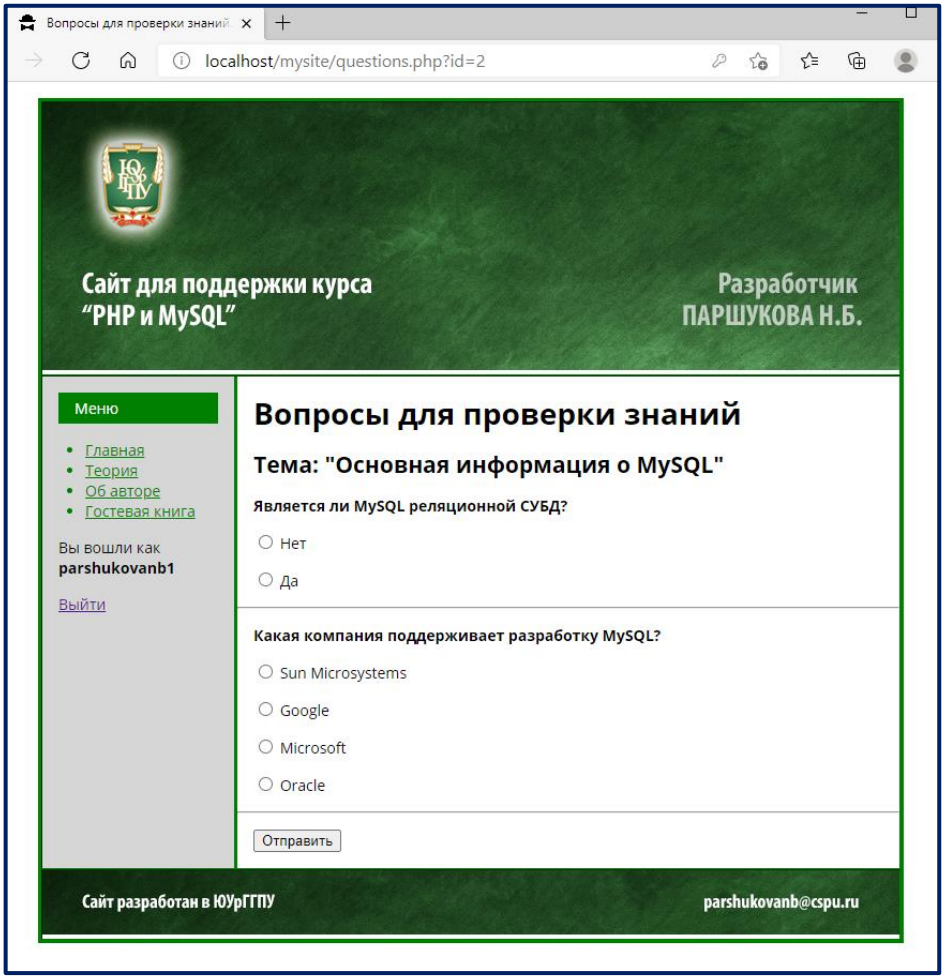


Рис. 79. Внешний вид теста для отдельной статьи

Задание 1. Создадим таблицу **questions** в нашей базе данных **my_db** с содержанием вопросов к имеющимся статьям. Пример заполненной таблицы указан на рисунке. Вам самим нужно определить тип поля, его длину, выявить ключевое поле и счетчик (рисунок 80).

id	article_id	quest	answers	correct
1	1	кто придумал язык PHP?	Стив Джобс|Билл Гейтс|Расмус Лердорф|Тим Бернерс Л...	3
2	1	Как правильно расшифровывается PHP?	Hypertext Preprocessor|Pages HTTP Personal|Planet ...	1
3	2	Является ли MySQL реляционной СУБД?	Нет|Да	2
4	2	Какая компания поддерживает разработку MySQL?	Sun Microsystems|Google|Microsoft|Oracle	4

Рис. 80. Пример контрольных вопросов в таблице questions

Здесь поле **article_id** – это номер статьи из таблицы **articles**, к которой в дальнейшем и будет закреплен данный вопрос.

Поле **answers** содержит перечисление вариантов ответа, отделенных друг от друга специальным разделителем – вертикальной чертой **|**.

Поле **correct** содержит порядковый номер правильного ответа. Например, для первого вопроса «Кто придумал язык PHP?» правильным ответом будет «Расмус Лердорф», стоящий под номером **3**.

Сделайте в таблице не менее 2-х записей по каждой статье (если статей у вас две, то не менее четырех вопросов должно быть в таблице). Должно быть заполнено каждое поле. В дальнейшем Вы сможете пополнить содержание БД.

Задание 2. Выведем данные из таблицы **questions** на страницу с тестом.

1. Сделайте переход по гиперссылке со страницы **article.php** на страницу **questions.php** следующим образом (рисунок 81):

```
    </td>
</tr>
<tr>
    <td>
        <table width="800" border="0" cellpadding="0" cellspacing="0">
            <tr>
                <td width="180" valign="top" class="left-column">
                    <!-- здесь будет меню -->
                    <?php include("menu.php");?>
                </td>
                <td width="620" valign="top">
                    <!-- здесь будет контент -->
                    <h1><?php echo $myrow["title"]?></h1>
                    <p>Дата создания:<br><b><?php echo $myrow["date_created"]?></b></p>
                    <p>Автор:<br><b><?php echo $myrow["author"]?></b></p>
                    <?php echo $myrow["text"]?>
                    <p><a href="questions.php?id=<?php echo $myrow["id"]?>">Перейти в вопросам</a></p>
                    <!-- конец контента -->
                </td>
            </tr>
        </table>
    </td>
```

Рис. 81. Добавление ссылки для перехода к вопросам

2. Создайте файл **questions.php**, пересохранив файл **article.php** (страница с выводом отдельной статьи).

Измените содержимое файла таким образом, чтобы все сведения о статье собирались бы в массив **$article** (рисунок 82).

```php
<?php include("db_connect.php");
    if (isset($_REQUEST["id"])) {
        $id = intval($_REQUEST["id"]);
        $sql = "SELECT * FROM articles WHERE id=$id";
        $result = mysql_query($sql);
        $article = mysql_fetch_array($result);
    }
?>
<!DOCTYPE html>
```

Рис. 82. Добавление ссылки для перехода к вопросам

3. В теге <title>…</title> выведите фразу (тема выводится в соответствии с выбранной статьей) – рисунок 83.

```
<title>Вопросы для проверки знаний. Тема: [redacted]</title>
```

Рис. 83. Содержимое тега <title>…</title>

В контентной области с помощью заголовков первого и второго уровней (соответственно **<h1>…</h1>** и **<h2>…</h2>**) добейтесь вывода следующих сообщений (рисунок 84):

Рис. 84. Заголовки для страницы с вопросами

4. В контентной области создайте код вывода на страницу вопросов из таблицы **questions**, соответствующих выбранной статье. На рисунке некоторые фрагменты скрыты, вам нужно их восстановить (рис. 85).

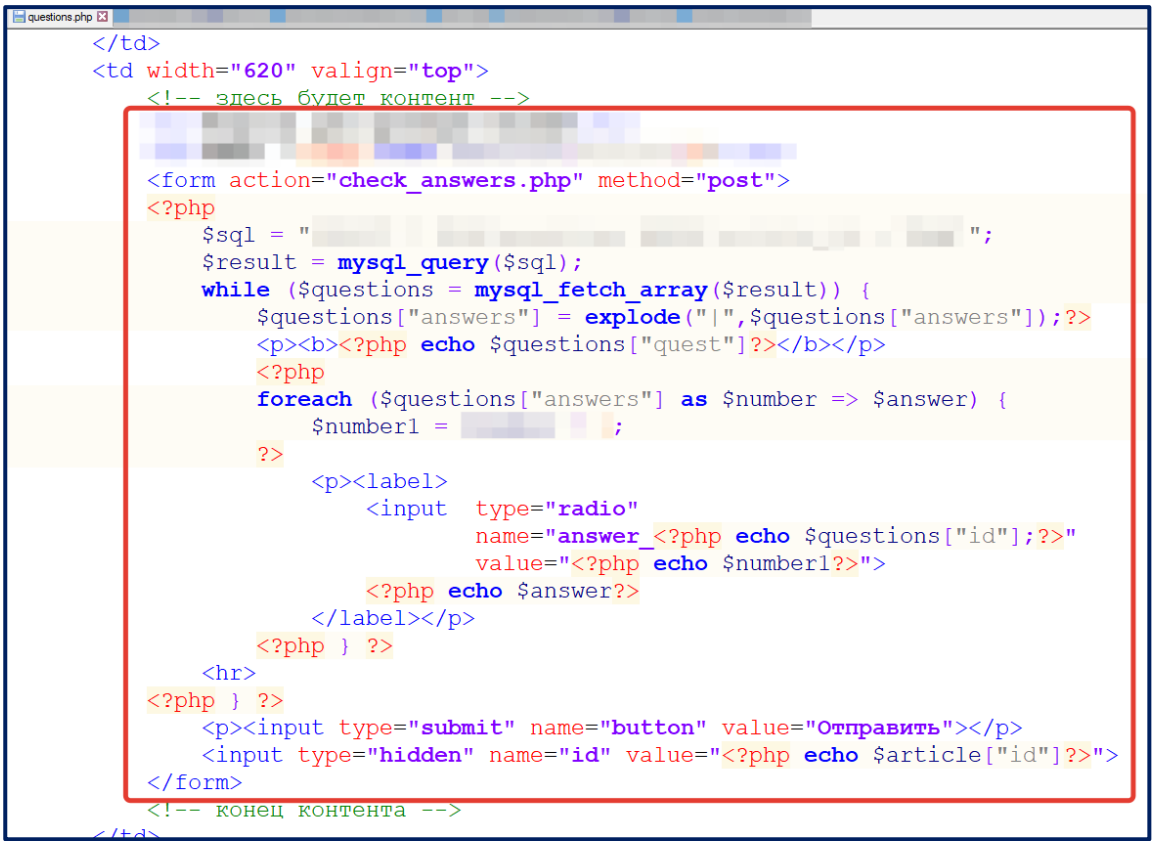


Рис. 85. Код вывода вопросов

Пояснение

В данном фрагменте использовалась функция **explode,** позволяющая разбить строку с помощью разделителя.

В нашем случае использовался разделитель «вертикальная черта» |

```
$questions["answers"] = explode("|",$questions["answers"]);
```

В результате работы этой функции мы получаем уже не строку, а массив, каждый элемент которого является отдельным вариантом ответа в нашем вопросе.

5. Проверьте работу программы: перейдите по ссылке «Теория» из главного меню, выберите одну из статей и перейдите на ее просмотр. Внизу должна быть ссылка «Перейти к вопросам». При переходе по ней вы должны увидеть страницу с вопросами для проверки знаний (рисунок 46).

Задание 3. Разработаем программу, которая проверит количество правильных ответов на тест.

1. Сделайте копию файла **questions.php** и новый файл назовите **check_answers.php**.

Добавьте код, показанный на рисунке, заполнив скрытые фрагменты программы. Объясните, что делает выделенный фрагмент программы (рисунок 86).

```php
<?php include("db_connect.php");
    if (isset($_REQUEST["id"])) {
        $id = intval($_REQUEST["id"]);
        $sql = "SELECT * FROM articles WHERE id=$id";
        $result = mysql_query($sql);
        $article = mysql_fetch_array($result);

        $sql = "[redacted]";
        $result = mysql_query($sql);
        $count = mysql_num_rows($result);
        $verno = 0;
        while ($questions = mysql_fetch_array($result)) {
            if ($questions["correct"] == $_POST["answer_".$questions["id"]]){
                $verno++;
            }
        }
        $itog=round(($verno*100)/$count,0);
    }
?>
<!DOCTYPE html>
<html>
    <head>
```

Рис. 86. Код для проверки правильных ответов

2. В теге **<title>…</title>** выведите фразу (тема выводится в соответствии с выбранной статьей) – рисунок 87.

```
<title>Результаты знаний. Тема: [redacted]</title>
```

Рис. 87. Содержимое тега <title>…</title> для страницы с результатами теста

3. В контентной части файла **check_answers.php** выведите результаты тестирования пользователя, заполнив пропущенные места. Для этого вам нужно внимательно изучить и прокомментировать код из задания 3 п. 1 (рисунок 88).

```
            </td>
        </tr>
        <tr>
            <td>
                <table width="800" border="0" cellpadding="0" cellspacing="0">
                    <tr>
                        <td width="180" valign="top" class="left-column">
                            <!-- здесь будет меню -->
                            <?php include("menu.php");?>
                        </td>
                        <td width="620" valign="top">
                            <!-- здесь будет контент -->
                            <h1><?php if ([redacted]) echo $login.", "?>ваши результаты </h1>
                            <h2>Тема: "<?php echo $article["title"]?>"</h2>
                            <p>Количество вопросов <b>[redacted]</b></p>
                            <p>Верных ответов <b>[redacted]</b></p>
                            <p>Результат <b>[redacted] %</b></p>
                            <!-- конец контента -->
                        </td>
                    </tr>
                </table>
```

Рис. 88. Вывод информации о тестировании

4. Проверьте работу всего алгоритма тестирования.

Устно ответьте на следующие вопросы:

- При каких ситуациях результаты будут соотнесены с авторизованным пользователем?
- Может ли неавторизованный пользователь пройти тестирование?
- Будут ли для неавторизованного пользователя выдаваться результаты?
- Каким образом в файл **check_answers.php** была передана переменная **id**, которая передает номер выбранной статьи?
- Объясните, почему в форме с вопросами использовался метод **POST**?

Задание 4. Дополнительное задание.

1. Дополните алгоритм файла **check_answers.php** таким образом, чтобы сохранялись результаты тестирования данного пользователя по данной статье. Для этого вам сначала нужно будет создать новую таблицу в базе данных, определиться с ее полями, их типами. Затем дописать алгоритм добавления нужных данных в таблицу (по аналогии с добавлением сообщения в гостевую книгу).

2. Проверьте работу программы. Покажите работу преподавателю.

3.7. Разработка фото галереи на сайте

Вопросы для повторения

1. Какой тег отвечают за вставку изображения на HTML-странице? Какие у этого тега есть атрибуты?

2. Какая HTML-форма может организовать загрузку файла с локального компьютера на сервер?

3. Какой атрибут тега <form> необходимо поставить, чтобы передать через форму файл?

4. Каким образом можно ограничить размер файла, передаваемый на сервер?

5. С помощью какой функции языка PHP происходит копирование файла на сервер?

Задачи лабораторной работы

- Создать таблицу базы данных для хранения изображений.
- Подготовить изображения и заполнить базу данных.
- Написать php-программу, выводящую изображения из базы данных на страницу/.
- Разработать админ-панель для редактирования изображений на сайте.

Задание 1. Разработайте таблицу **photos** в базе данных для хранения изображений. Необходимой информацией для изображений является: название изображения, имя файла изображения, дата загрузки изображения, опубликовано изображение или нет (поле **published**, тип поля **tinyint**, длина **1**).

Задание 2. Подготовьте 5-6 jpg-изображений и перенесите их в папку **photos** на вашем сайте. Создайте записи в таблице **photos** с информацией об этих изображениях (название изображения и т.д.). Значение поля **published** (опубликовано) установите равным единице.

Задание 3. Напишем php-программу, выводящую изображения из базы данных на страницу. Конечный вид будет примерно следующим – рисунок 89.

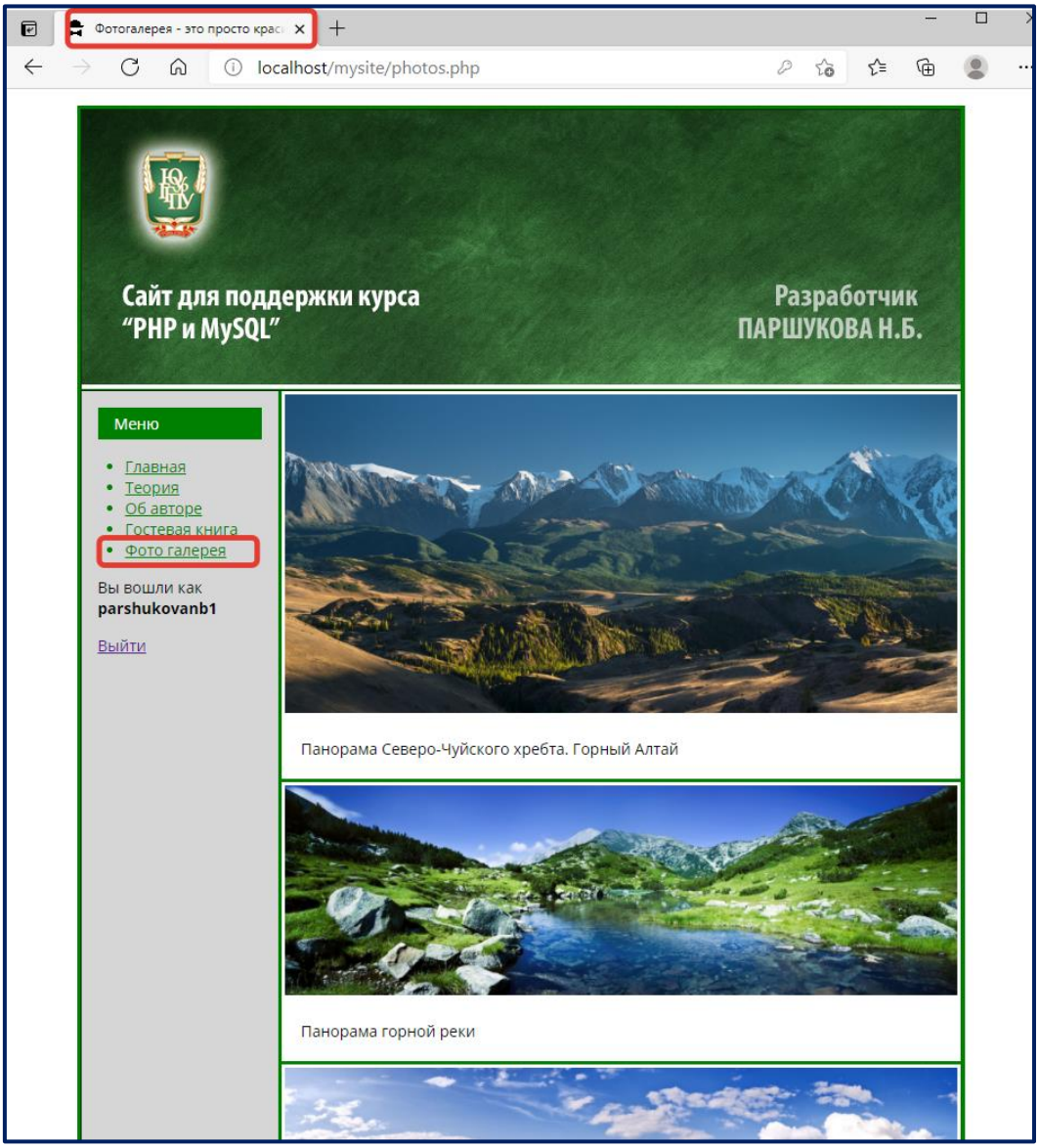


Рис. 89. Общий вид фотогалереи

1. В главном меню вашего сайта (файл **menu.php**) поставьте гиперссылку на вызов файла **photos.php** с будущей фотогалереей.

2. Создайте файл **photos.php** на основе ранее созданного файла с шаблоном вывода отдельной статьи (**article.php**).

3. Разработайте запрос, который будет выводить всю информацию об опубликованных изображениях (изображения, у которых значение поля **published** равно 0 не должны выводиться на это странице). Разработайте внешний вид для вывода фотографий. Рядом с фотографией должно выводиться ее название. В качестве атрибута **alt** тега изображений должен выводиться такой же тест, как и в названии фото (рисунок 90).

```
<table width="800" border="0" cellpadding="0" cellspacing="0">
    <tr>
        <td width="180" valign="top" class="left-column">
            <!-- здесь будет меню -->
            <?php include("menu.php");?>
        </td>
        <td width="620" valign="top">
            <!-- здесь будет контент -->
            <?php
                $sql = "                                                    ";
                $result = mysql_query($sql);
                while ($myrow = mysql_fetch_array($result)) {
            ?>
                <table width="100%" class="article">
                    <tr>
                        <td>
                            <img src="photos/<?php echo $myrow["filename"]?>" alt="<?php echo $myrow["name"]?>" width="100%">
                        </td>
                    </tr>
                    <tr>
                        <td>
                            <p><?php echo $myrow["name"]?></p>
                        </td>
                    </tr>
                </table>
            <?php } ?>
            <!-- конец контента -->
```

Рис. 90. Код формирования фотогалереи

Проверьте выполнение алгоритма. Попробуйте поставить для одной записи в таблице значение поля **published** равным 0. Что изменилось в вашей выдаче?

Задание 4. Разработаем админ-панель для редактирования фотогалереи. Для этого:

1. в корне вашего сайта разместите папку **admin**. Все php-программы, имеющие отношение к панели администрирования, мы будем размещать в ней. Скопируйте следующие файлы и папки из корня вашего сайта в папку **admin**.

- index.php
- style.css
- images
- db_connect.php
- menu.php
- photos.php.

2. Отредактируйте файл **admin/index.php** и **admin/menu.php** так, чтобы пользователю было понятно, что он находится в админ-панели (рис. 58). Обратите внимание, что в файле **menu.php** осталась одна ссылка, ведущая на файл **photos.php**, который будет отвечать у нас за отображение таблицы редактирования фотогалереи (рисунок 91).

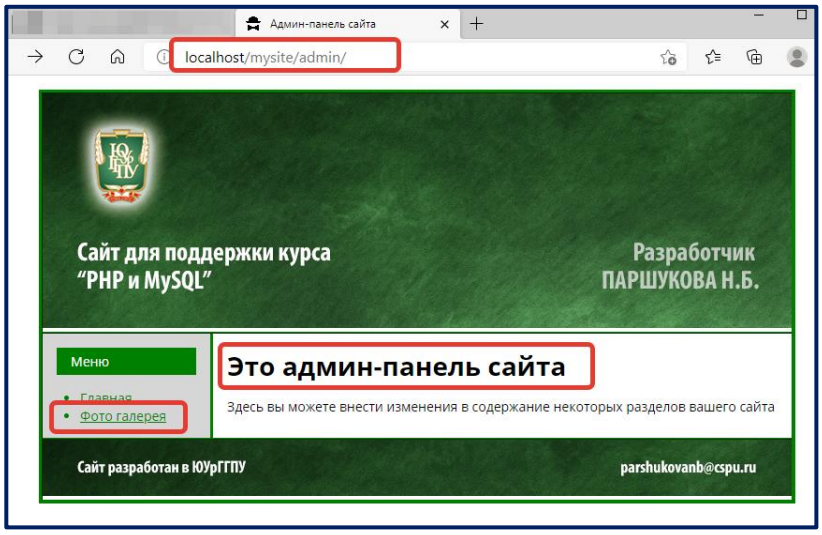


Рис. 91. Главная страница панели администратора

3. Измените файл **photos.php**, который будет выводить таблицу с нашей фотогалереей, таким образом, чтобы ее вид соответствовал рисунку 92.

id	Название	Изображение	Дата загрузки			
4	Камчатка		2021-05-23 20:46:32	⊗	✓	✎
3	Белорецк горы Южный Урал Башкирия		2021-05-23 20:46:47	⊗	✗	✎
2	Панорама горной реки		2021-05-23 20:47:01	⊗	✓	✎
1	Панорама Северо-Чуйского хребта. Горный Алтай		2021-05-23 20:47:11	⊗	✓	✎

Рис. 92. Вид фотогалереи в админ-панели

Вы можете скопировать код для вывода фотогалереи. Однако следует учесть, что в админ-панели должны выводиться абсолютно все изображения: и опубликованные и неопубликованные (рис. 93).

photos.php

```
<!-- здесь будет контент -->
<table width="100%" class="big-table">
    <tr>
        <td>id</td>
        <td>Название</td>
        <td>Изображение</td>
        <td>Дата загрузки</td>
        <td></td>
        <td></td>
        <td></td>
    </tr>
<?php
    $sql = "SELECT * FROM photos ORDER BY date_created";
    $result = mysql_query($sql);
    while ($myrow = mysql_fetch_array($result)) {
        $link_delete = "photo_handler.php?task=delete&id=".$myrow["id"];
        $task = "publish";
        $img_published = "uncheck.png";
        if ($myrow["published"]){
            $task = "unpublish";
            $img_published = "check.png";
        }
        $link_publish = "photo_handler.php?task=".$task."&id=".$myrow["id"];
        $link_edit    = "photo_handler.php?task=edit&id=".$myrow["id"];
?>
        <tr>
            <td><?php echo $myrow["id"]?></td>
            <td><?php echo $myrow["name"]?></td>
            <td><img src="../photos/<?php echo $myrow["filename"]?>" width="200px"></td>
            <td><?php echo $myrow["date_created"]?></td>
            <td><a href="<?php echo $link_delete?>">
                    <img src="images/delete.png" alt="Удалить изображение" width="30px">
                </a></td>
            <td><a href="<?php echo $link_publish?>">
                    <img src="images/<?php echo $img_published?>" width="30px">
                </a></td>
            <td><a href="<?php echo $link_edit?>">
                    <img src="images/edit.png" alt="Редактировать изображение" width="30px">
                </a></td>
        </tr>
<?php } ?>
</table>
<!-- конец контента -->
```

*Рис. 93. Код вывода таблицы с информацией об изображениях **my-site/admin/photos.php***

Пояснение

Поясним в вышеприведенном коде некоторые моменты. По аналогии с гостевой книгой, нами будет создан файл **photo_handler.php**, который возьмет на себя обработку задач с нашими изображениями. Он будет удалять изображения, делать их опубликованными или неопубликованными, редактировать названия, загружать новые файлы и др. Пока в приведенном коде заложена возможность удалить изображение и сделать его опубликованным / неопубликованным.

Для отображения состояния «опубликовано/не опубликовано» нужно подобрать два изображения, например,  – опубликовано (имя файла **check.png**),  – не опубликовано (имя файла **uncheck.png**). Определите по коду самостоятельно, в какой папке должны храниться данные изображения.

Для удаления записи о файле мы будем отображать пиктограмму  (имя файла в коде **delete.png**).

Для редактирования записи будем отображать пиктограмму  (имя файла **edit.png**)

4. Создайте файл **photo_handler.php**, который будет осуществлять всю основную работу по редактированию информации в фотогалерее. Задача этого файла – получать данные с форм, обрабатывать эти данные, формировать запросы к базе данных и осуществлять принудительный редирект (перенаправление) на нужный файл с шаблоном. Файл **photo_handler.php** будет решать следующие задачи:

- Изменение состояния: опубликовано или нет.
- Удаление записи о файле (и физическое удаление файла с сервера).
- Создание новой записи о файле в таблице базы данных и редактирование выбранной записи (у них примерно один алгоритм, поэтому их объединим).
- Сохранение информации о новом файле или об отредактированном файле.

Каждая задача будет передаваться в файл **photo_handler.php** через переменную **task**. Обратите внимание, что на рисунке 92 все ссылки, ведущие на файл **photo_handler.php** сопровождаются **GET** параметром **task**.

4.1. Первая задача, которая будет решена с помощью нашего файла **photo_handler.php,** будет изменение состояния записи о файле – опубликовано или нет. Кроме переменной **$task** мы будем получать и переменную **$id**, которая будет хранить **id** изменяемой записи.

Самостоятельно прокомментируйте код функции **publishPhoto** (рисунок 94).

photo_handler.php

```php
<?php
    include("db_connect.php");
    if (isset($_GET["task"]))    $task = $_GET["task"];
    if (isset($_REQUEST["id"])) $id = $_REQUEST["id"];
    switch ($task) {
        case "publish":
        case "unpublish":
            publishPhoto($task);
            break;
    }

    function publishPhoto($task){
        $id = intval($_REQUEST["id"]);
        $published = 1;
        if ($task == "unpublish") $published = 0;
        $sql = "UPDATE photos SET published = '$published' WHERE id = '$id'";
        $result     = mysql_query($sql);
        if ($result) {
            header("location:photos.php");
        }
    }
```

*Рис. 94. Код файла **photo_handler.php***

4.2. Вторая задача – удаление изображения. Здесь следует отметить, что удалить выбранную запись из базы данных – мало. Необходимо физически удалять изображение с сервера. Если этого не сделать, то со временем произойдет переполнение сервера такими не удалёнными изображениями. Поэтому разобьем наш алгоритм на две части: удаление файла с сервера и удаление информации из базы данных.

Допишите в файл **photo_handler.php** функцию **deletePhoto**. Запрос на удаление записи вам нужно сделать самостоятельно. Также самостоятельно создайте в начале файла **photo_handler.php** ветку case со значением «delete» (по аналогии с *publish / unpublish*) – рисунок 95**.**

```
32      function deletePhoto($id){
33          removePhotoFromServer($id);
34          $result     = mysql_query("[illegible]");
35          if ($result) {
36              header("location:photos.php");
37          }
38      }
```

Рис. 95. Функция для удаления фото

В данном коде идет вызов функции **removePhotoFromServer()** (удалить изображение с сервера), код которой также нужно разместить в файле **photo_handler.php** (функция **removePhotoFromServer()** вспомогательная, поэтому для нее ветку case делать не нужно) – рисунок 96**.**

```
77      function removePhotoFromServer($id){
78          $sql = "SELECT filename FROM photos WHERE id = '$id'";
79          $result = mysql_query($sql);
80          $filename = mysql_fetch_array($result);
81          $filename = $filename["filename"];
82          if ($filename && file_exists('../photos/'.$filename)){
83              unlink('../photos/'.$filename);
84          }
85      }
```

Рис. 96. Функция для удаления изображения с сервера

Проверьте функцию удаления изображений с сервера. Обратите внимание, что удаляется и фото из папки **mysite/photos**. Для правильного обращения к ней был использован путь **../photos.** Две точки означают переход на родительскую директорию по отношению к текущей директории. Родительской директорией по отношению к папке **admin** (именно в ней находится **photo_handler.php**) является **mysite**. Значит, формально обращение идет **mysite/photos**. Пути с двумя точками называется относительными.

4.3. Третья задача, которую будет решать наш обработчик **photo_handler.php**, – будет редактирование текущей или добавление новой записи. Одна и та же функция будет решать две задачи. Поэтому ей нужно передавать значение выбранной задачи **$task**, чтобы при редактировании текущей записи получать данные из базы (рисунок 97).

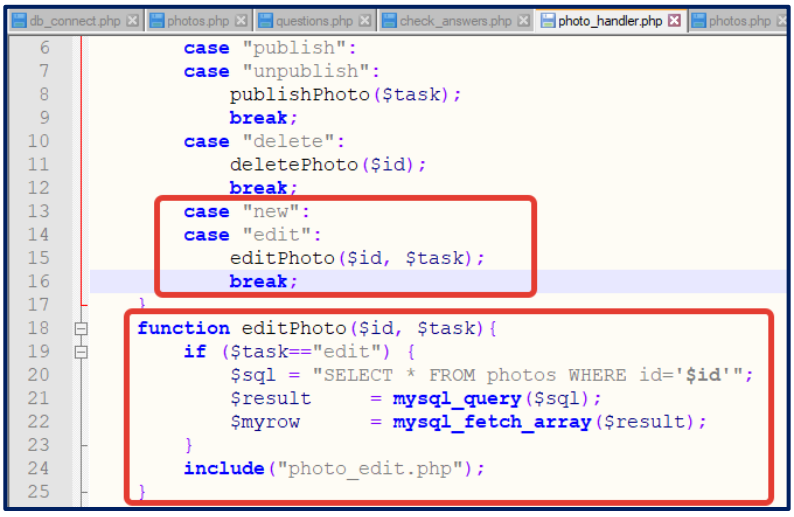


Рис. 97. Функция для редактирования фото

Также в этом коде подключается файл с шаблоном редактирования отдельной записи о файле **photo_edit.php**. Создайте копию файла **photos.php** и назовите новый файл **photo_edit.php**. В этот файл будет передаваться массив **$myrow**, который будет содержать данные по запросу текущей записи о файле. Если же запись будет новой, то массив **$myrow** будет пуст и пользователю нужно будет внести информацию о файле (рисунок 98).

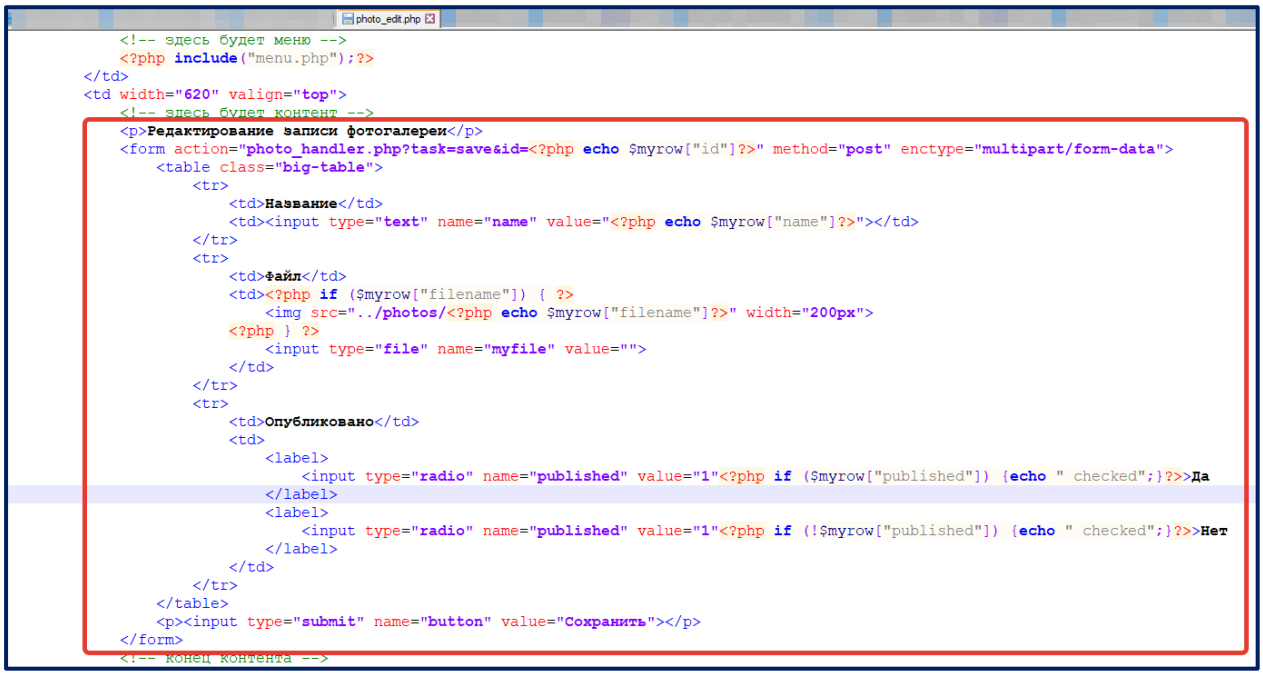


Рис. 98. Шаблон для редактирования фото

Проверьте созданный шаблон следующим образом: на странице со списком всех фотографий http://localhost/mysite/admin/photos.php нажмите на  рядом с одним из изображений. Вы должны оказаться примерно на такой странице (рисунок 99):

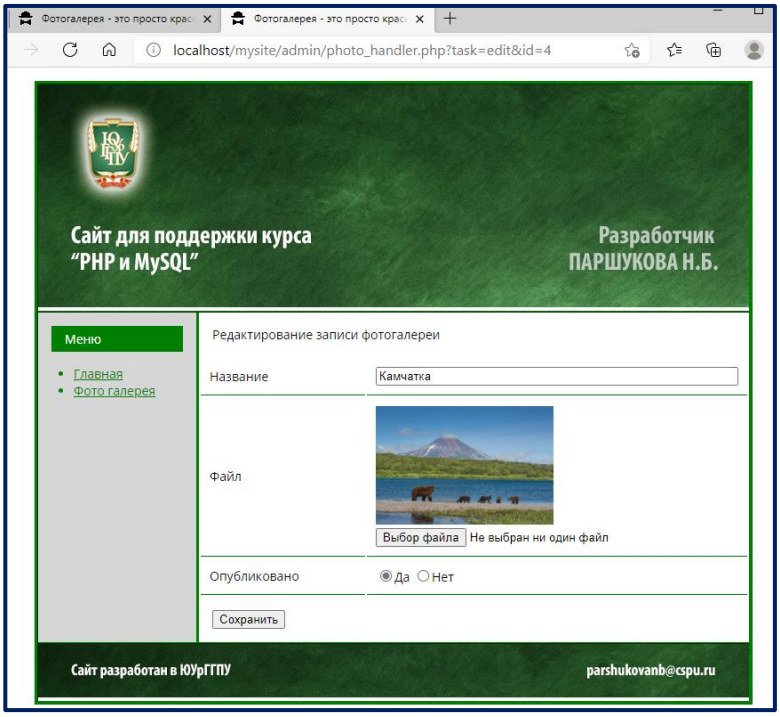


Рис. 99. Вид страницы редактирования фото

Дополним наш алгоритм возможностью создавать новые записи о файлах. Для этого в файле **mysite/admin/photos.php** разместите гиперссылку для добавления новой записи (рис. 100):

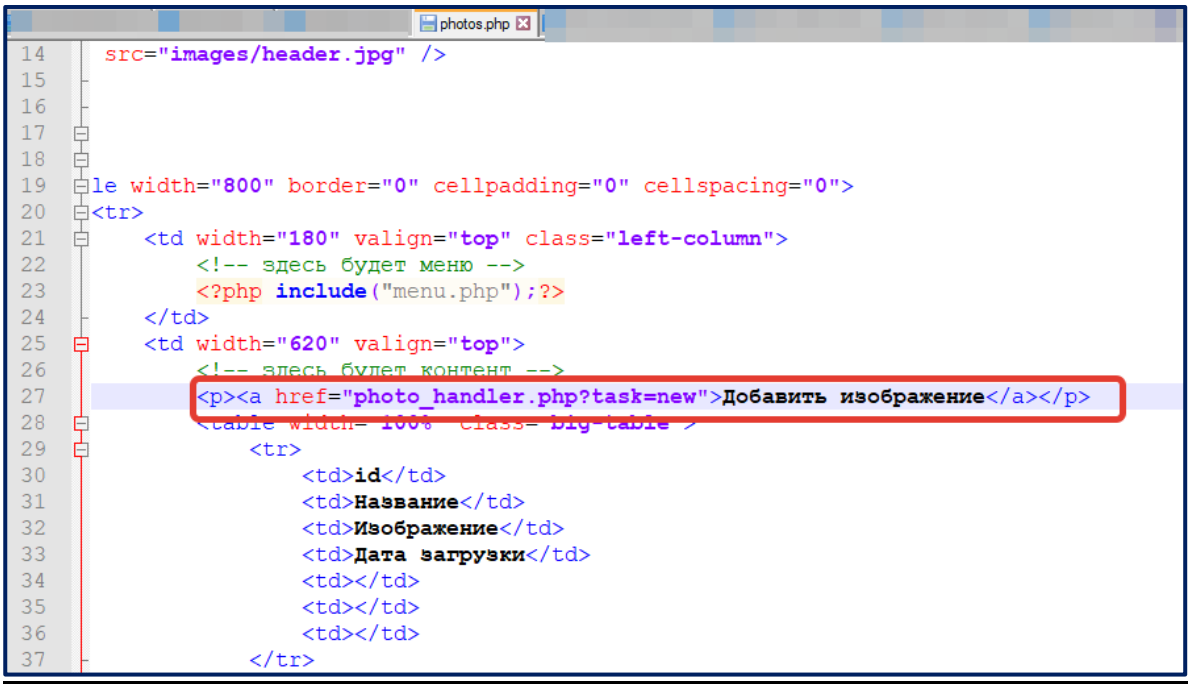


Рис. 100. Гиперссылка для добавления нового фото

Проверьте, что при нажатии на эту гиперссылку открывается такой же шаблон для редактирования записи, но все поля пустые (рисунок 101).

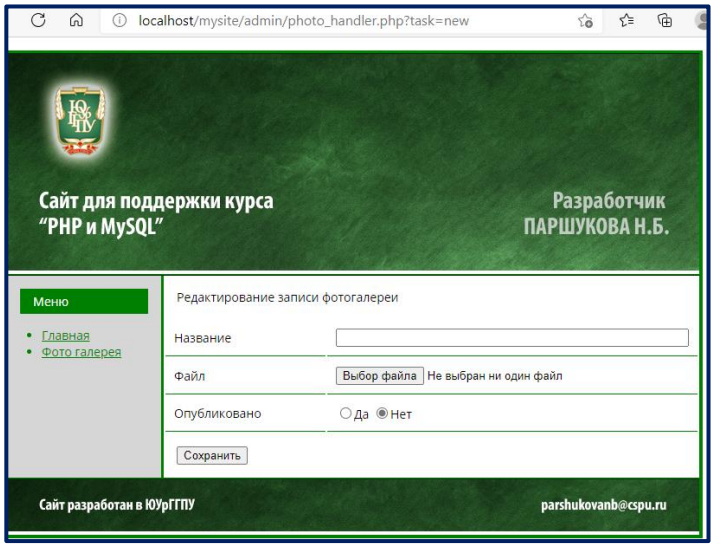


Рис. 101. Вид шаблона для добавления нового фото

4.4. Четвертой задачей является возможность сохранить информацию о файле. В файле **photo_edit.php** обработчиком для формы был указан получатель (рисунок 102):

```
<!-- здесь будет контент -->
<p>Редактирование записи фотогалереи</p>
<form action="photo_handler.php?task=save&id=<?php echo $myrow["id"]?>" method="post" enctype="multipart/form-data">
    <table class= bly-table >
        <tr>
            <td>Название</td>
            <td><input type="text" name="name" value="<?php echo $myrow["name"]?>"></td>
```

Рис. 102. Обработчик формы

В файле **photo_handler.php** создадим вариант с выбором **case “save”** для оператора **switch**. Допишите строку (рисунок 103).

```
case "save":
    savePhoto($id);
    break;
```

Рис. 103. Новая строка оператора switch для вызова функции

Создайте в файле **photo_handler.php** функцию **savePhoto** (рисунок 104).

```
function savePhoto($id){
    $name       = addslashes($_POST["name"]);
    $published  = addslashes($_POST["published"]);
    $file       = $_FILES["myfile"];
    if ($id) {
        $sql = "UPDATE photos SET name = '$name', date_created = NOW(), published = '$published' WHERE id = '$id'";
        $result     = mysql_query($sql);
    } else {
        $sql = "INSERT INTO photos (name,date_created,published) VALUES ('$name',NOW(),'$published')";
        $result     = mysql_query($sql);
        $id = mysql_insert_id();
    }
    uploadPhotoToServer($file, $id);
    header("location:photos.php");
}
```

*Рис. 104. Код функции **savePhoto***

Пояснение

В этом коде из формы нужно получить методом POST все переменные – имя (**$name**), публикация (**$published**), файл (**$file**). Файл получаем из суперглобального массива **$_FILES**, атрибут **myfile** соответствует названию поля для ввода файла.

Если переменная **$id** была передана при сохранении (ее значение не 0), то мы должны обновить текущую запись, соответственно запрос на **UPDATE**.

Если переменная **$id** равна 0, то запись новая, и все данные нужно вставить в таблицу с помощью запроса **INSERT**. При этом в результате запроса будет создана запись с новым значением **id**, которое можно получить с помощью функции **mysql_insert_id**.

Отдельно нужно поработать с файлом. Новый файл нужно физически загрузить на сервер, при этом удалив старый. Поэтому логика работы с файлом здесь вынесена в отдельную функцию **uploadPhotoToServer**

В файле photo_handler.php создайте функцию загрузки файла на сервер **uploadPhotoToServer**. Обратите внимание, что на сервер загружаются фото только определенных расширений (jpg, png, gif). При успешной загрузке информация о загруженном файле сохраняется в таблице **photos** (рисунок 105).

```
function uploadPhotoToServer($file, $id){
    if ($file["name"]){
        $extensions = array("jpg","png","gif","JPG","PNG","GIF");
        $ext = explode(".",$file["name"]);
        $ext = $ext[count($ext)-1];
        if (in_array($ext, $extensions)){
            if (move_uploaded_file($file["tmp_name"], "../photos/".$file["name"])){
                removePhotoFromServer($id);
                $sql = "UPDATE photos SET filename = '".$file["name"]."' WHERE id = '$id'";
                $result     = mysql_query($sql);
            }
        }
    }
}
```

*Рис. 105. Код функции **uploadPhotoToServer***

Проверьте работу алгоритма сохранения изменений. Покажите работу преподавателю.

3.8. Публикация сайта на удаленном сервере

Завершающим этапом создания сайта является выбор площадки хостинга и регистрация домена.

Итак, в данной лабораторной работе мы зарегистрируем для сайта домен и разместим его на удаленном хостинге.

Для размещения выберем бесплатный хостинг **https://beget.com/**.

1. Зайдите на сайт https://beget.com/ru/free-hosting и зарегистрируйтесь на нем. Внимание, хостинг требует подтверждения по телефону (рисунок 106).

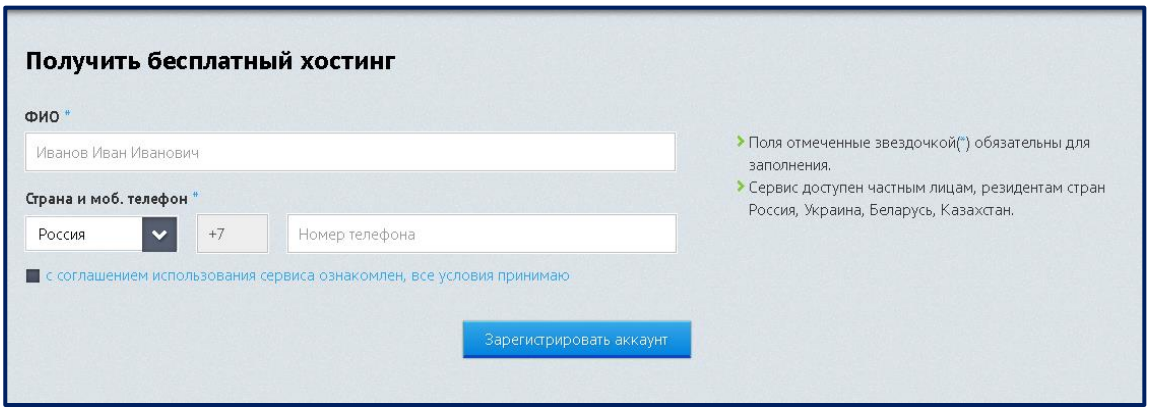


Рис. 106. Страница регистрации пользователя на хостинге beget.com

2. После подтверждения по телефону у вас появится информация о логине и пароле на хостинг (рисунок 107). Обязательно сохраните ее.

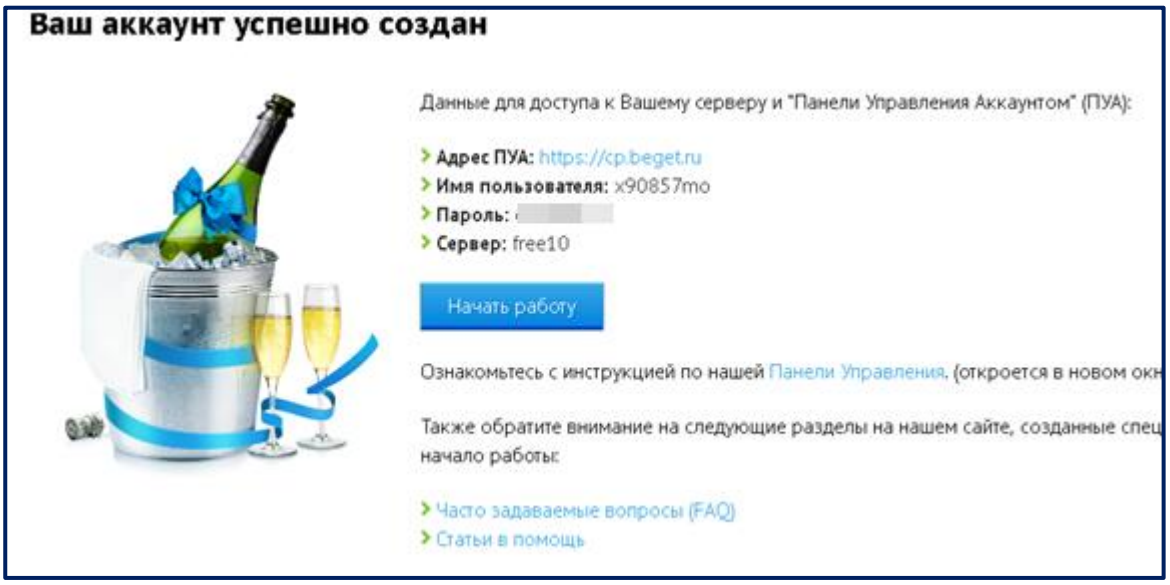


Рис. 107. Страница со сгенерированными логином и паролем

3. Перейдите в панель управления (ПУА) вашим сайтом.

4. Перенос на хостинг нашего сайта будет состоять из двух этапов: копирование файлов сайта и копирование базы данных.

Копирование сайта. Заархивируйте свой сайт в **zip** архив. Ваш сайт находится по адресу home/localhost/www/mysite. В архив добавляйте только **содержимое** папки **mysite**. Метод сжатия не принципиален (рисунок 108).

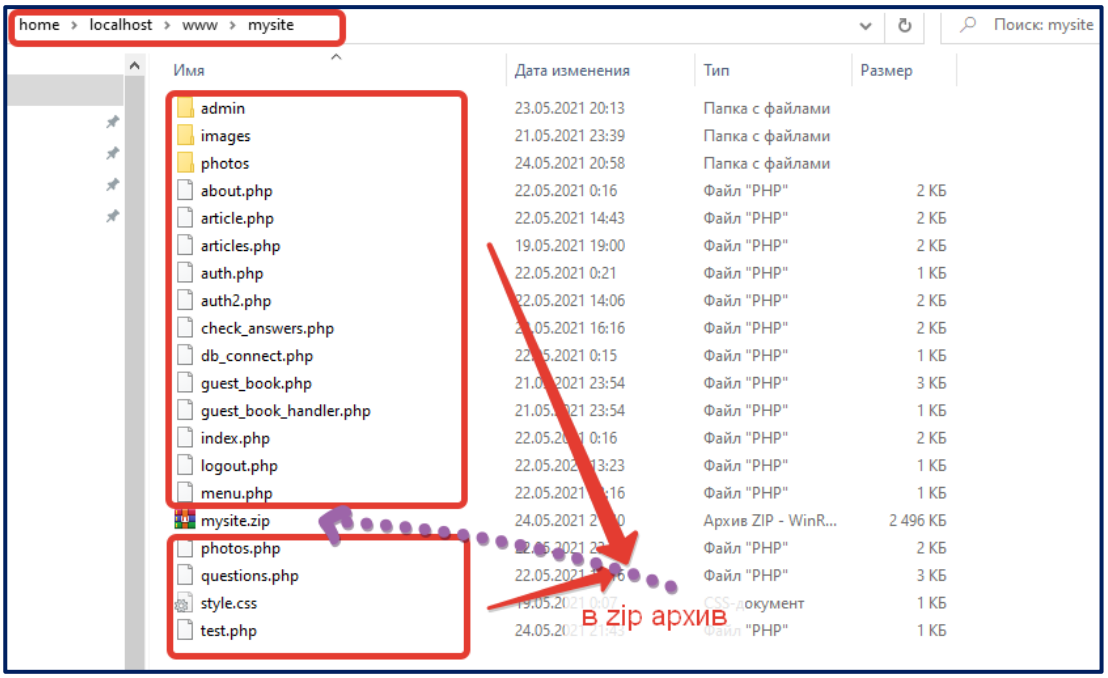


Рис. 108. Архивация всех файлов сайта

5. Перейдите в файловый менеджер панели управления вашего хостинга на beget.ru. Выберите папку с названием вашего аккаунта, примерно таким **x90857mo.bget.ru**, где **x90857mo** – имя пользователя, выданное вам при регистрации на beget.ru. Внутри нее должна быть папка **public_html.** Сюда скопируйте архив с вашим сайтом wordpress (рисунок 109).

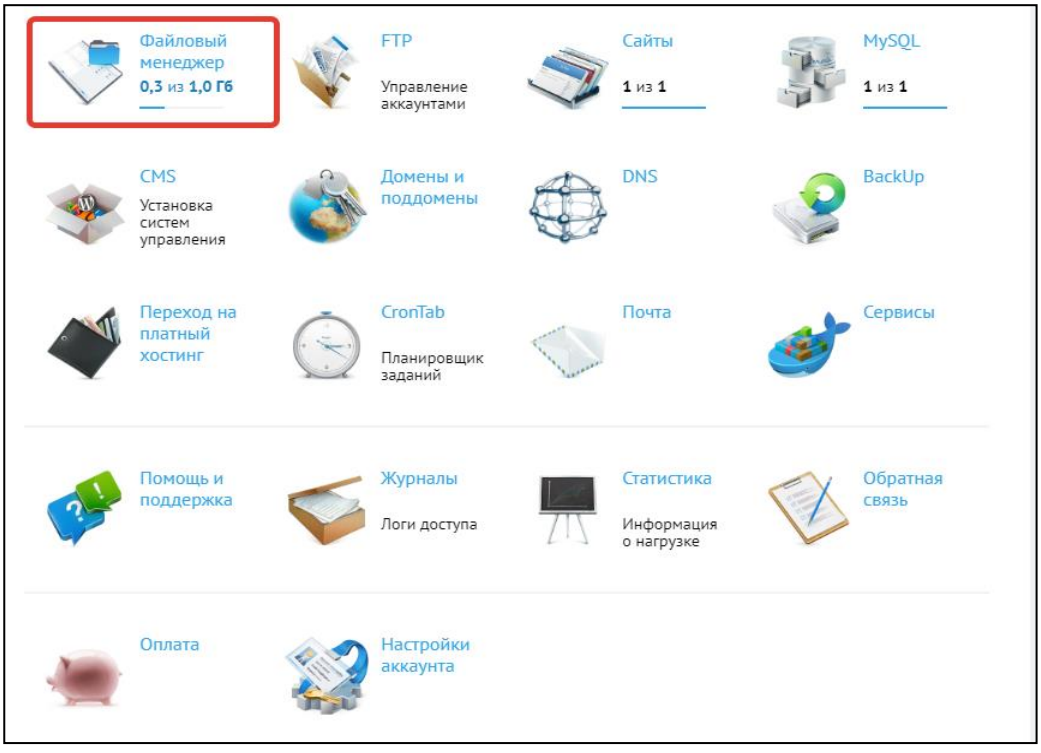
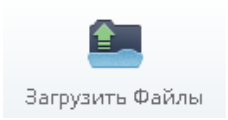


Рис. 109. Вид панели управления хостингом beget.com

6. С помощью кнопки **Загрузить файлы** 

выполните загрузку файлов на сервер. Распакуйте архив (правой кнопкой мыши по архиву / **Распаковать**) – рисунок 110.

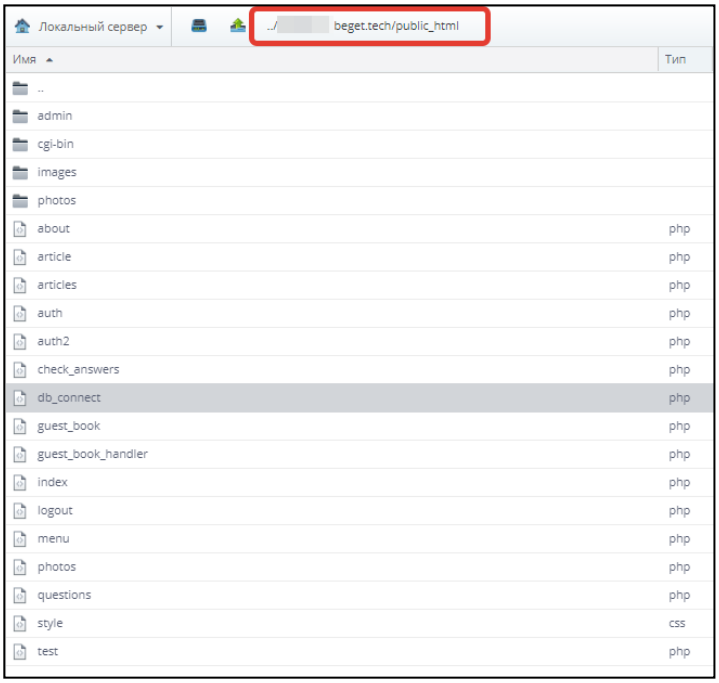


Рис. 110. Вид файлов сайта на хостинге

7. Далее создадим базу данных для нашего сайта. С помощью утилиты phpmyAdmin сделаем копию (backup) нашей базы данных. В браузере введите адрес http://localhost/tools/phpmyadmin. Откроется страница утилиты. В левом фрейме выберите базу данных **my_db**. Примерный вид показан на рисунке 111.

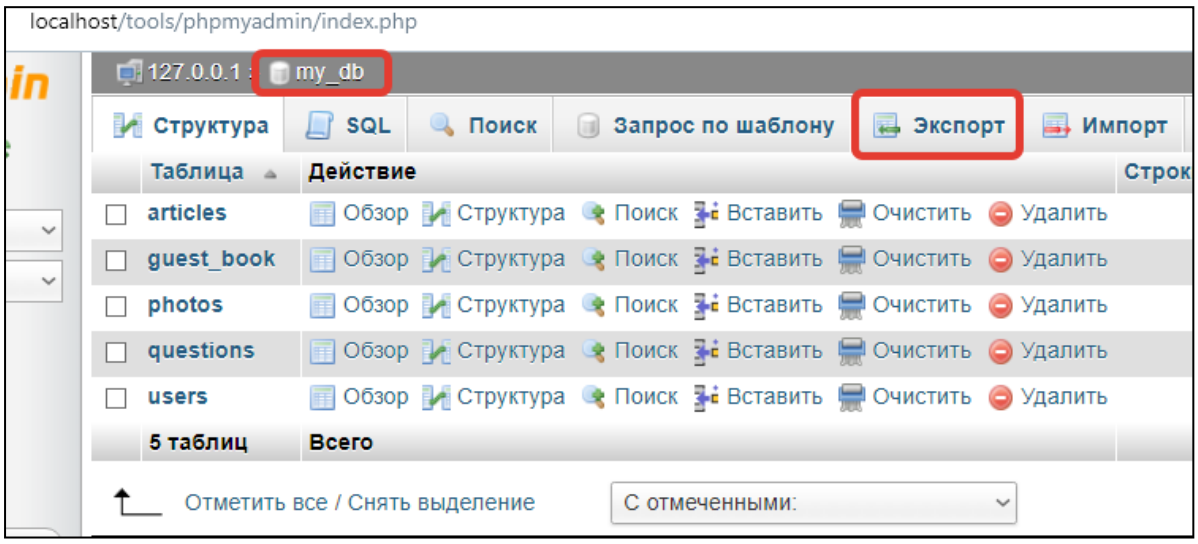


Рис. 111. Вид утилиты phpMyAdmin и кнопки «Экспорт»

Перейдите на вкладку Экспорт. Далее установите следующие настройки экспорта:

- Способ экспорта – **обычный**;
- Компрессия – нет (способ сжатия);
- Добавить выражение **DROP TABLE / VIEW / PROCEDURE / FUNCTION** – галочка.

Остальные параметры оставить без изменения. Нажать на кнопку **OK** и сохранить файл на компьютере.

При экспорте будет создан файл **my_db.sql**, который, скорей всего, будет сохранен в папке **Загрузки** на вашем компьютере (у некоторых браузеров папка для загрузки файлов может отличаться).

8. Сделаем импорт базы данных на хостинг. Для этого перейдите в панель управления хостингом https://cp.beget.com/main. С помощью кнопки **MySQL** (рисунок 112) создадим новую базу данных. Для удобства сделайте совпадающими имя базы данных и имя пользователя. Нажмите **«Создать»**.

Рис. 112. Создание базы данных MySQL для сайта

Обязательно запишите себе имя пользователя и придуманный вами пароль. Он понадобится для подключения базы данных к сайту.

После создания БД появится ссылка для перехода в **phpmyadmin** на хостинге (рисунок 113). Перейдите по ней.

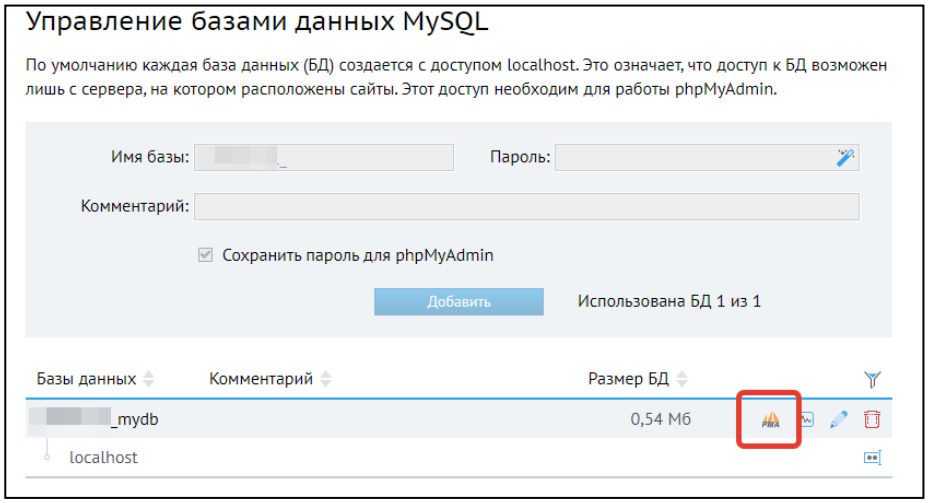


Рис. 113. Управление базами данных MySQL в панели управления хостингом

9. Импортируйте архивный файл своей базы данных (вкладка **Импорт** в phpmyAdmin) – рисунок 114.

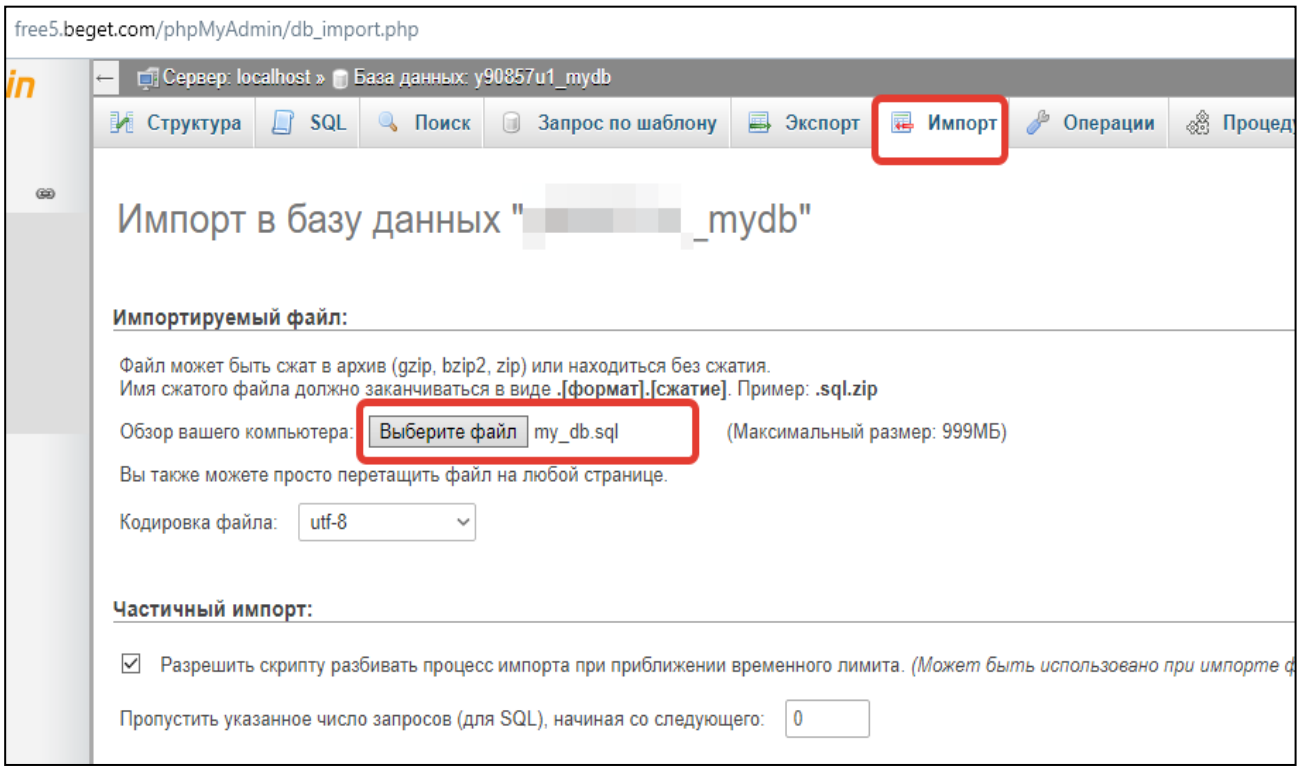


Рис. 114. Импорт базы данных на сервер

10. Зайдите снова в файловый менеджер в панели управления хостингом **beget.com.** Откройте на редактирование файл **db_connect.php** и измените данные для подключения к базе данных (имя базы данных и пароль вы должны были сохранить в отдельном файле, рисунок 115).

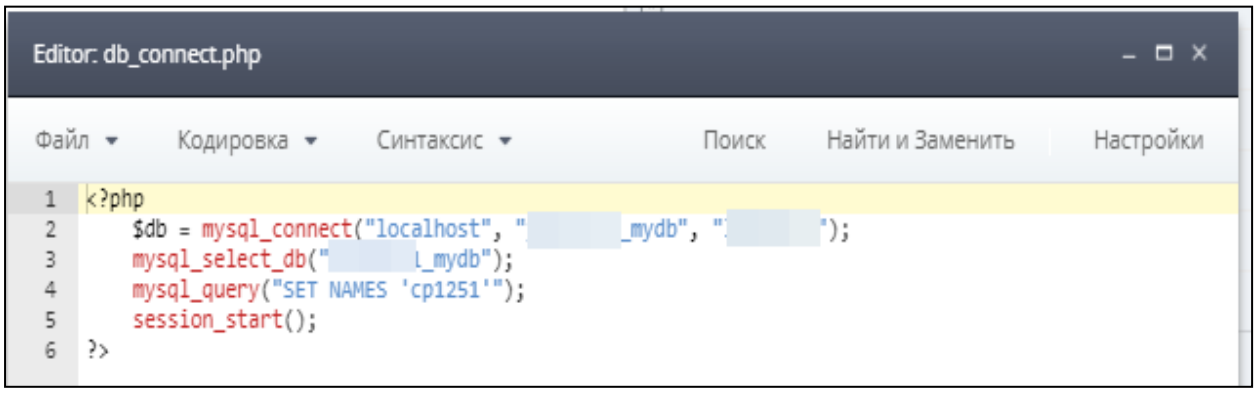


Рис. 115. Редактирование файла wp-config.php

11. Проверьте работу вашего сайта. Адрес у всех будет разный, но его можно увидеть в панели управления вашим сайтом. Указанный адрес скопируйте в адресную

строку на новой вкладке браузера. Также следует проверить и админ-панель по работе с фотографиями (к адресу добавить **/admin**) – рисунок 116.

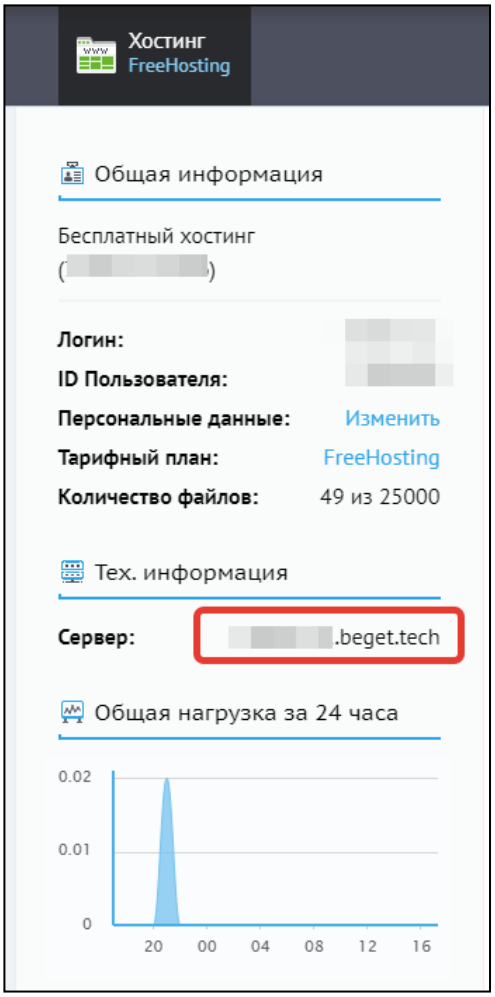


Рис. 116. Получение адреса сервера

ГЛАВА 4. РЕШЕНИЕ НЕКОТОРЫХ ПРАКТИЧЕСКИХ ЗАДАЧ СРЕДСТВАМИ PHP И MYSQL

Глубокое понимание алгоритмизации и успешное обучение программированию возможно только при должной степени самостоятельности обучающегося. Безусловно, такую сложную предметную область как программирование осваивать легче с преподавателем. Но преподаватель должен организовать самостоятельную работу студента, руководить ею и наблюдать за ее динамикой [8]. Одним из эффективных способов погружения в предметную область разработчика является реализация проектов [13]. Так студент сталкивается с прикладными задачами, реализуя собственный проект. В данной главе рассмотрены некоторые типовые задачи, реализация которых будет дополнять учебный проект студента, и в то же время эти задачи требуют большей самостоятельности по сравнению с лабораторными работами.

4.1. Поиск на сайте

Поиск по содержимому – важная функция сайта. Все мы знаем, как удобно работают поисковые системы Яндекс, Google и другие. Однако их поиск осуществляется только по страницам, проиндексированным этими поисковыми системами. В случае, если сайт имеет большое количество статей и другого контента, он должен иметь собственный поиск. В этом параграфе мы рассмотрим реализацию поиска на сайте.

Для начала вспомним, с чего начинается поиск информации – в специальную поисковую форму пользователи вбивают поисковый запрос, нажимают на кнопку «Найти», и через считанные секунды у них перед глазами краткие фрагменты найденной информации. Дальше уже сам пользователь принимает решение, на какой из сайтов переходить. Конечно, поисковые системы Яндекс и Google используют сложные алгоритмы ранжирования (сортировку, упорядоченность сайтов), умеют понимать сложную морфологию языков и многое другое. В нашем случае мы сделаем достаточно простой алгоритм поиска. Но он будет достаточно эффективным, потому что поиск будет осуществляться по ограниченному контенту нашего сайта.

Предположим, что в нашей базе данных, в таблице **articles** имеется достаточно большое количество статей. По каким полям будет осуществляться поиск? Текстовы-

ми полями в нашей таблице являются заголовок (title), краткое описание (description), полное описание (text), автор статьи (author). Пользователь может захотеть найти информацию как по фрагменту заголовка, автору или фрагменту в любой части статьи.

Сам поиск будет осуществляться с помощью SQL-запроса. Роль PHP в данном случае – получить от пользователя поисковый запрос, проверить его длину, если длина не слишком маленькая (больше 4-х символов), то сформировать и отправить поисковый запрос в базу данных. Затем, уже получив данные из базы, оформить их для представления пользователю. Поисковый запрос на выборку будет использовать условие **LIKE** («похожий»), который позволяет найти подстроку в строке (см. п. 2.4).

Представление результатов поиска у нас уже фактически имеется – это шаблон для вывода статей. Наличие уже готового кода значительно облегчит нам работу.

Добавьте вызов файла **search.php** в файл для вывода меню – **menu.php** – как это показано на рисунке 117.

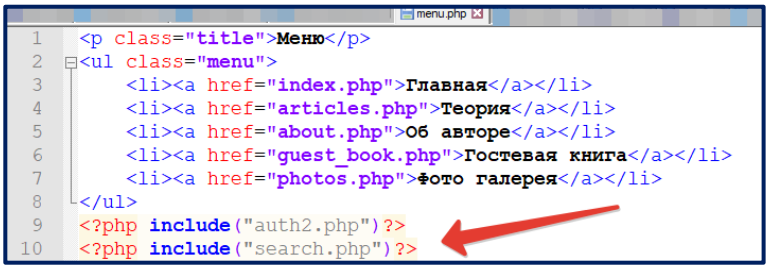


Рис. 117. Добавление вызова файл search.php из menu.php

Создадим файл **search.php** в папке с нашим сайтом. В этом файле будет находиться поисковая форма, состоящая из текстового поля с именем **searchword** и кнопки «Найти» (рисунок 118).

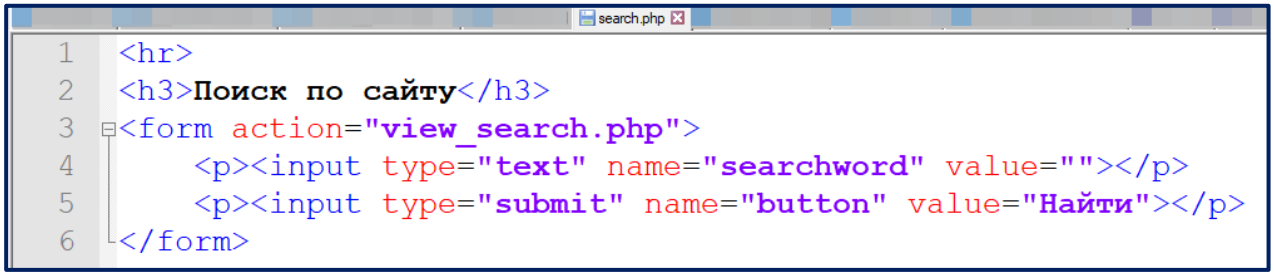


Рис. 118. Код файла search.php

Обработчиком данной формы является скрипт **view_search.php**, который можно сделать очень быстро – достаточно скопировать содержимое файла **articles.php**, выводящего список статей.

Внесем только некоторую проверку: если пользователь не нажал на кнопку «Найти», то будем считать такой поисковый запрос не валидным, и в этом случае пользователь должен получать сообщение: «Вы обратились к поиску без ввода данных». Если же длина поискового запроса будет менее 4-х символов, то будем считать такой запрос также не валидным, т.к. в этом случае может найтись слишком большое количество результатов. Также перед результатами поиска будем подсчитывать, сколько было найдено результатов.

Код файла **view_search.php** представлен на рисунке 119.

view_search.php

```
<!-- здесь будет контент -->
<?php
    if (isset($_GET["searchword"])) $searchword = $_GET["searchword"];
    if (isset($_GET["button"]))     $button = $_GET["button"];
    if (!$button) { ?>
        <p>Вы обратились к поиску без ввода данных</p>
    <?php } else {
        if (strlen($searchword)<4) { ?>
            <p>Поисковый запрос не введен или его длина меньше 4-х символов</p>
    <?php } else {  ?>
        <h2>По запросу "<?php echo $searchword?>" найдено:</h2>
    <?php
            //удаляем лишние пробелы
            $searchword = trim($searchword);
            //добавляем экранирование \ для спецсимволов
            $searchword = stripslashes($searchword);
            //преобразует некоторые HTML теги в HTML сущности (преобразует в код)
            $searchword = htmlspecialchars($searchword);
            $sql = "SELECT * FROM articles WHERE title LIKE '%$searchword%'
                    OR description LIKE '%$searchword%' OR text LIKE '%$searchword%' OR author LIKE '%$searchword%'";
            $result = mysql_query($sql);
            $count  = mysql_num_rows($result);
            echo "<p>Найдено: <b>$count</b> статей</p>";
            while ($myrow = mysql_fetch_array($result)) {
            ?>
                <table width="100%" class="article">
                    <tr>
                        <td width="250">
                            <p><a href="article.php?id=<?php echo $myrow["id"]?>"><?php echo $myrow["title"]?></a></p>
                            <p>Дата создания:<br><b><?php echo $myrow["date_created"]?></b></p>
                            <p>Автор:<br><b><?php echo $myrow["author"]?></b></p>
                        </td>
                        <td>
                            <?php echo $myrow["description"]?>
                        </td>
                    </tr>
                </table>
            <?php }  ?>
    <?php } ?>
<?php } ?>
<!-- конец контента -->
```

Рис. 119. Код файла view_search.php

Конечный результат выглядит следующим образом – рисунок 120.

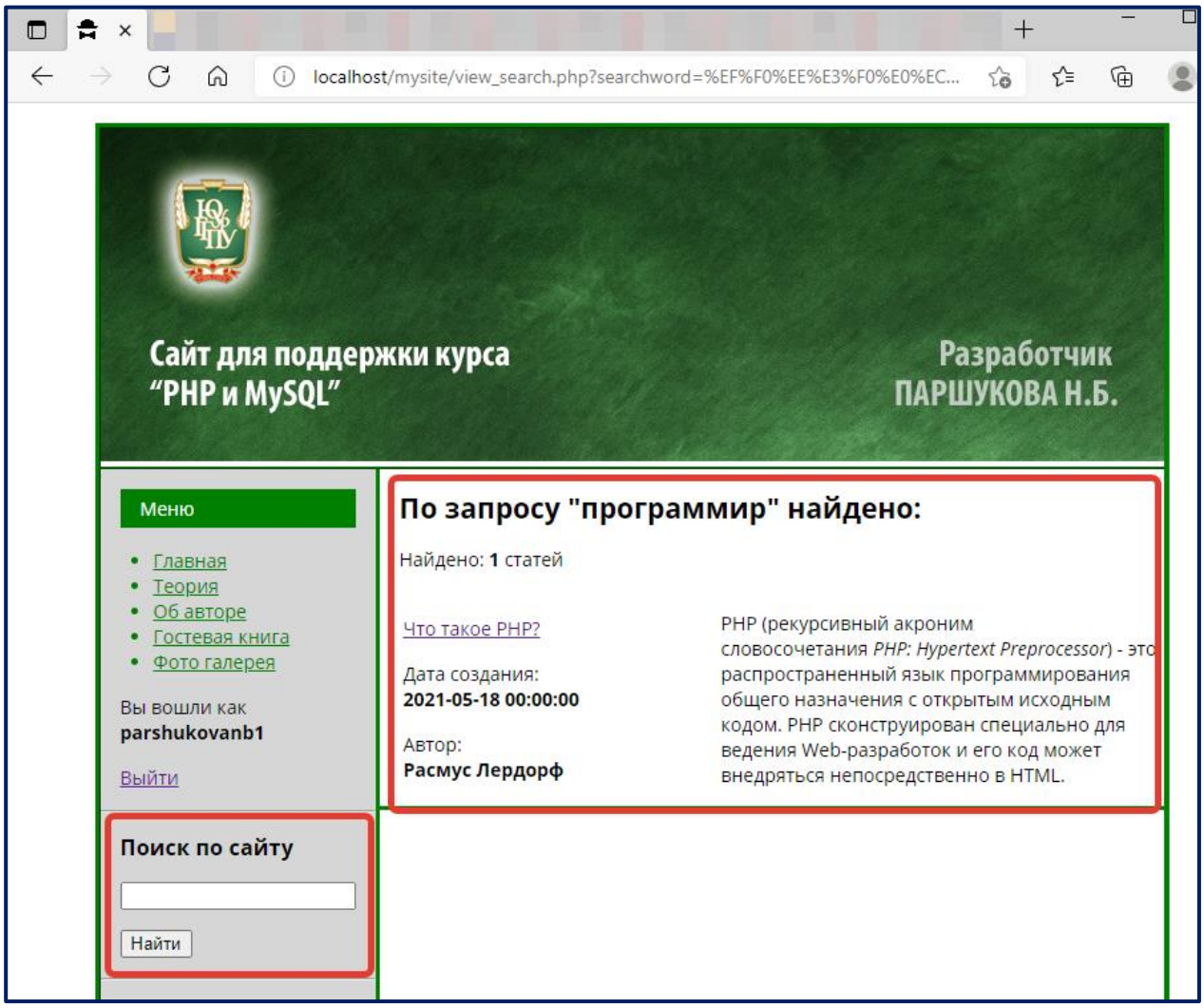


Рис. 120. Конечный результат поискового алгоритма

4.2. Вывод статистических данных на сайте

Рассмотрим создание блока для вывода статистических данных. С использованием агрегатных функций MySQL можно вывести общее количество статей, средний балл за тест группы студентов или самую последнюю дату в списке комментариев. В любом случае, блок статистики на сайте принесет полезную информацию для посетителей сайта.

Конечный результат блока вывода статистических данных будет выглядеть как показано на рисунке 121. Он будет располагаться после блока с поиском.

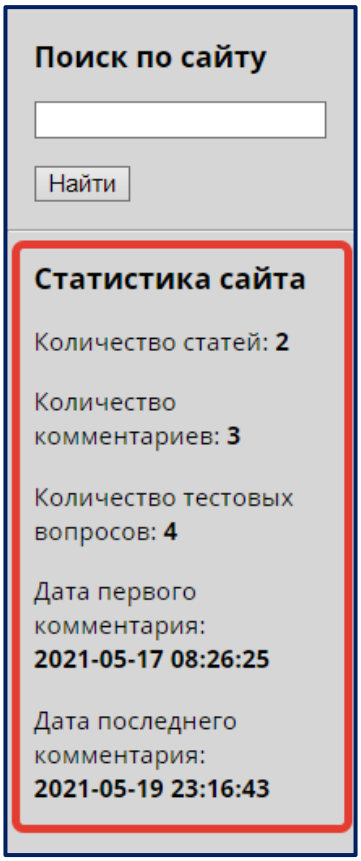


Рис. 121. Блок вывода статистики на сайте

Добавим вызов файла **stat.php** в файле с выводом меню **menu.php**, аналогично как это было сделано для подключения формы поиска (рисунок 117).

```
<?php include("stat.php")?>
```

Создайте файл **stat.php**, который расположите в корне вашего сайта.

Так как большинство агрегатных запросов очень похожи друг на друга, их можно оформить в виде функции, в которую передаются два параметра: поле с агрегатным запросом (**$field**) и имя таблицы, в которой отыскивается это значение (**$table**).

```
function getStatFromTable($field, $table){
    $sql = "SELECT $field FROM $table";
    $result = mysql_query($sql);
    $count = mysql_fetch_array($result);
    return $count[0];

}
```

Вызов этой функции может осуществляться следующим образом:

```
$count_articles = getStatFromTable("COUNT(*)", "articles");
```

Общий код файла **stat.php** показан на рисунке 122.

```php
<?php
    function getStatFromTable($field, $table){
        $sql = "SELECT $field FROM $table";
        $result = mysql_query($sql);
        $count = mysql_fetch_array($result);
        return $count[0];
    }

    $count_articles     = getStatFromTable("COUNT(*)", "articles");
    $count_comments     = getStatFromTable("COUNT(*)", "guest_book");
    $count_questions    = getStatFromTable("COUNT(*)", "questions");
    $min_data_comments  = getStatFromTable("MIN(date_created)", "guest_book");
    $max_data_comments  = getStatFromTable("MAX(date_created)", "guest_book");
?>
<hr>
<h3>Статистика сайта</h3>
<p>Количество статей: <b><?php echo $count_articles?></b></p>
<p>Количество комментариев: <b><?php echo $count_comments?></b></p>
<p>Количество тестовых вопросов: <b><?php echo $count_questions?></b></p>
<p>Дата первого комментария: <br><b><?php echo $min_data_comments?></b></p>
<p>Дата последнего комментария: <br><b><?php echo $max_data_comments?></b></p>
```

Рис. 122. Код файла stat.php

Попробуйте составить еще 2-3 запроса с использованием вычисления среднего арифметического, суммы, минимального и максимальных значений.

4.3. Создание опросов на сайте

Разработаем опрос. Для него понадобится две таблицы: первая – для хранения самих опросов, а вторая – для хранения результатов ответов на опросы.

На рисунке 123 показана таблица **polls** для хранения опросов. Определить типы и атрибуты полей этой таблицы можно уже самостоятельно. Также необходимо заполнить эту таблицу (по крайней мере 2 записи нужно внести).

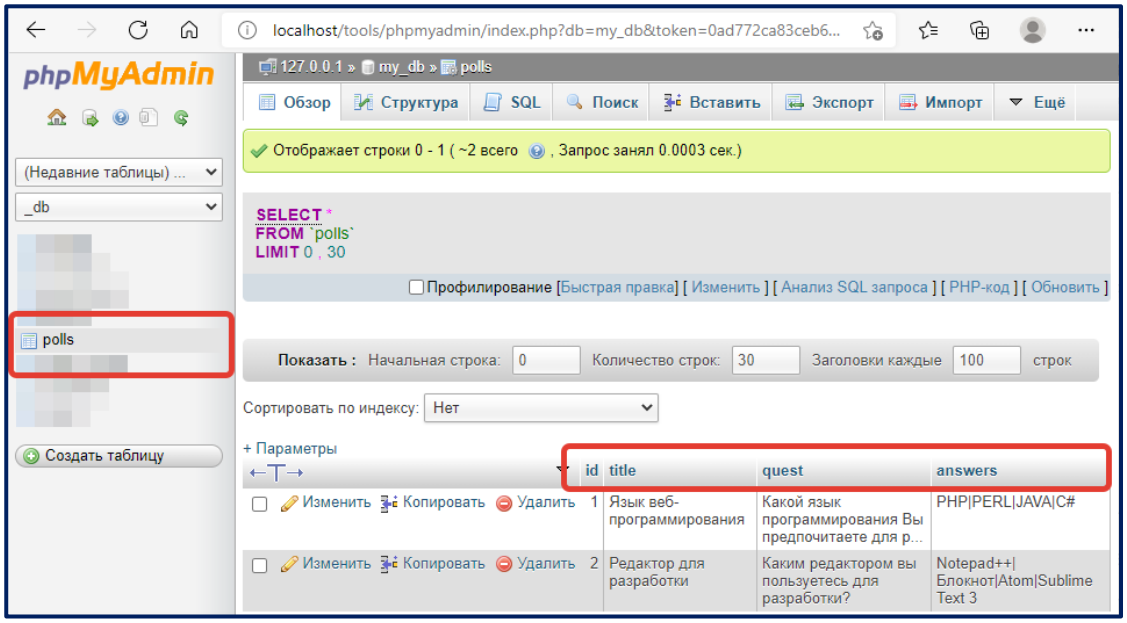


Рис. 123. Структура таблицы polls

На рисунке 124 показана таблица **polls_results** для хранения результатов опросов. Типы и атрибуты полей можно задать самостоятельно. Вставлять значения в эту таблицу не нужно.

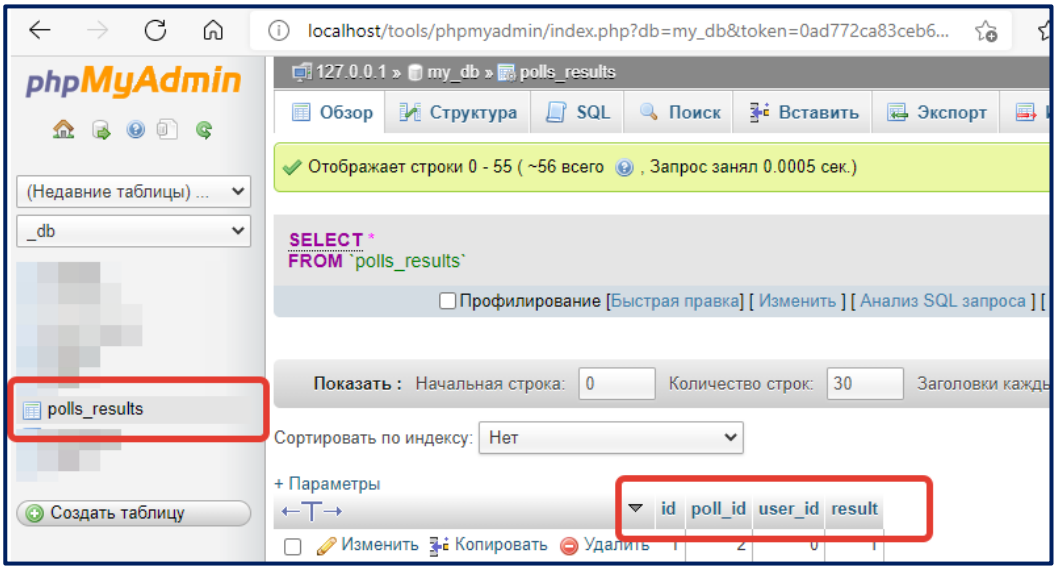


Рис. 124. Структура таблицы polls_results

Алгоритм проведения опроса будет следующим. Пользователь переходит на страницу опросов **polls.php**, на которой ему предлагается выбрать опрос для голосования или сразу же перейти на просмотр результатов по определенному опросу (рисунок 125).

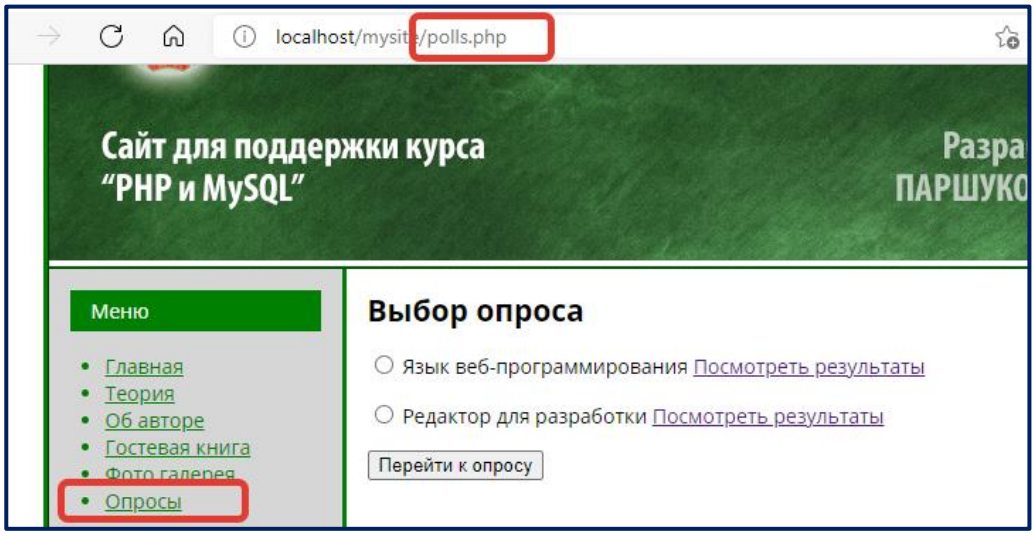


Рис. 125. Гиперссылка на страницу Опросы

Если пользователь выбрал с помощью радиоточки нужный опрос и нажал на кнопку «Перейти к опросу», то ему предлагается ответить на вопросы. Отметим, что с помощью механизма cookie нужно реализовать проверку, отвечал ли этот пользователь на данный опрос или нет. При этом, если пользователь уже отвечал на опрос, нужно запретить ему повторно

отвечать в течение какого-то времени (например, в течение часа, суток и т.д.). Вариантов реализации этого механизма множество, в нашем случае мы не будем запрещать пользователю принимать участи в опросе повторно, однако запоминать его повторный результат не будем. Об этом пользователя обязательно предупредим сообщением (рисунок 126).

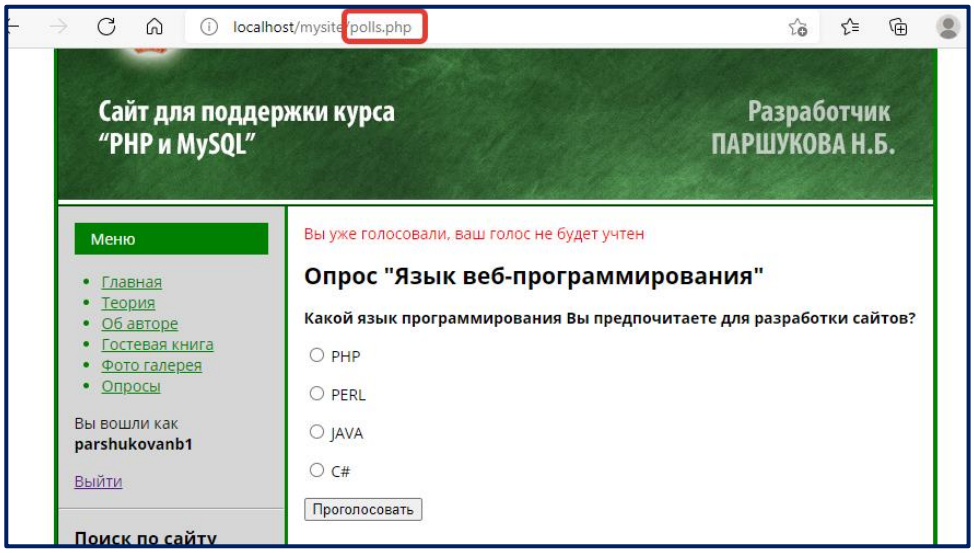


Рис. 126. Внешний вид скрипта polls.php

После того как пользователь проголосовал, наш алгоритм будет перенаправлять пользователя на все ту же страницу **polls.php**, но уже с GET параметром **view_poll_id**, содержащим номер опроса для просмотра результатов. Подсчет количества ответов за тот или иной вариант будем осуществлять с помощью агрегатной функции COUNT, при этом важно определить и результаты в процентах. С помощью тега **<hr>** (горизонтальная черта) можно сделать очень простую диаграмму – длина тега <hr> будет составлять вычисленный нами процент (рисунок 127).

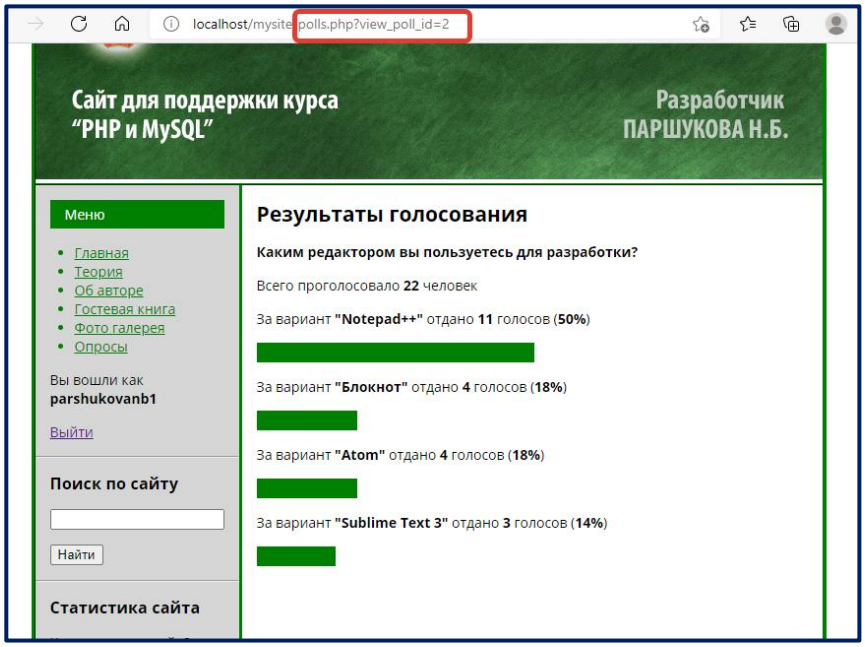


Рис. 127. Внешний вид страницы с результатами голосования

Ход работы

1. Для начала добавьте ссылку на файл **polls.php** в файл для вывода меню **menu.php**. Создайте файл **polls.php** (можно сделать файл-копию уже ранее созданного articles.php, как это уже делалось неоднократно).

2. В контентной области **polls.php** разместите следующий код (рисунок 128) – он будет подгружать запросы и представления для трех разных ситуаций:

- **view_polls.php** – выводит список опросов;
- **view_poll.php** – выводит форму для проведения голосования по выбранному опросу;
- **view_poll_results.php** – выводит результаты голосований по отдельному опросу.

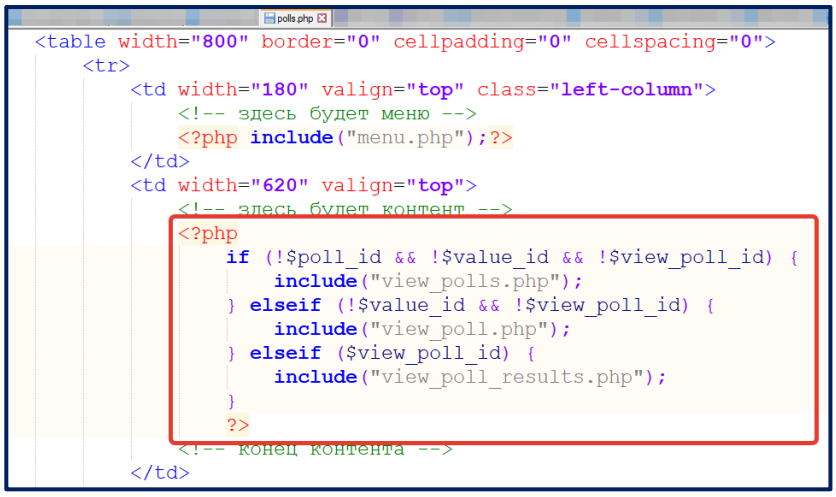


Рис. 128. Алгоритм вывода polls.php

3. Далее создадим код файла **view_polls.php** (рисунок 129). Как видно из приведенного кода, делается запрос к таблице **polls** и выводятся все опросы в виде радиоточек. Рядом располагаются гиперссылки с переходом на страницу просмотра результата.

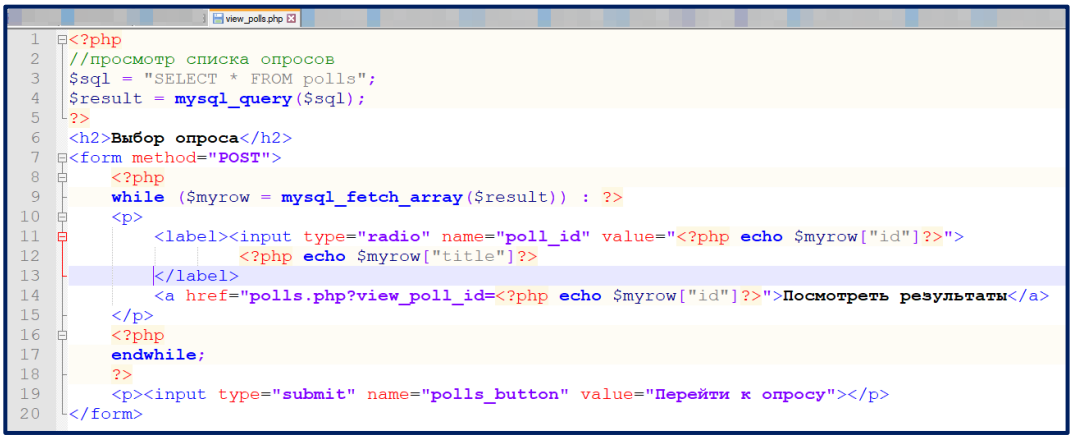


Рис. 129. Содержимое файла view_polls.php

4. Алгоритм файла **view_polls.php** (рисунок 130) формирует выборку по заданному параметру **$poll_id** из таблицы **polls**. Выводит варианты в виде радиоточек. Сами варианты ответов хранятся в поле answers в виде строки с разделителем «|». Функция **explode()** разбивает такую строку на массив элементов, чем удобно пользоваться в циклах. Параметр **$str_voted** хранит сообщение, если пользователь уже голосовал (его мы зададим позднее). Через форму передается скрытое поле **poll_id**, чтобы при просмотре результатов опроса **view_poll_results.php** данный параметр также был передан.

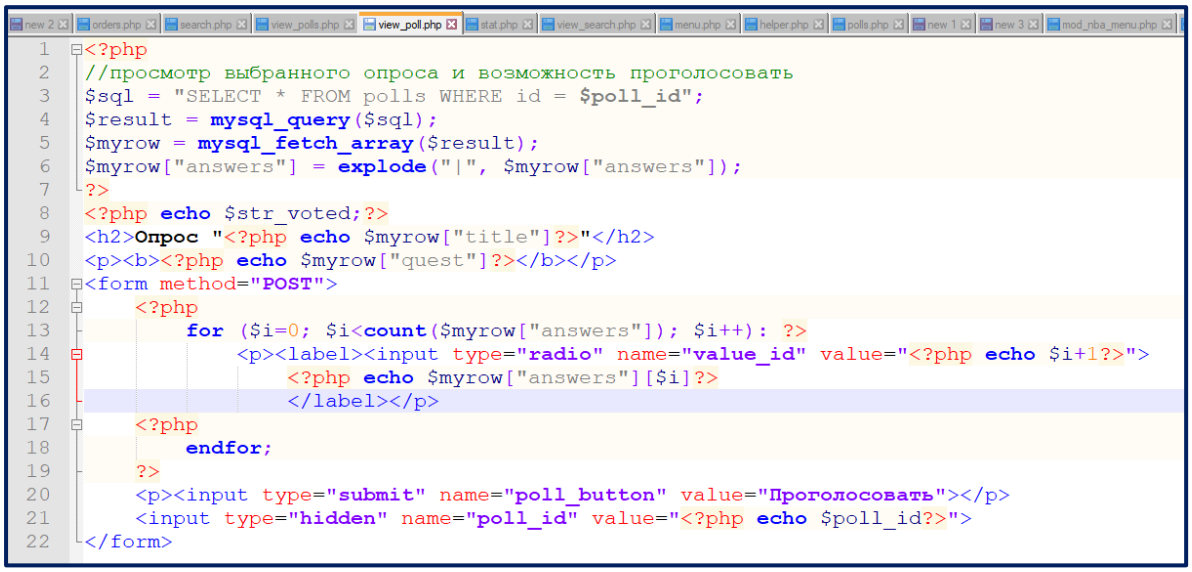
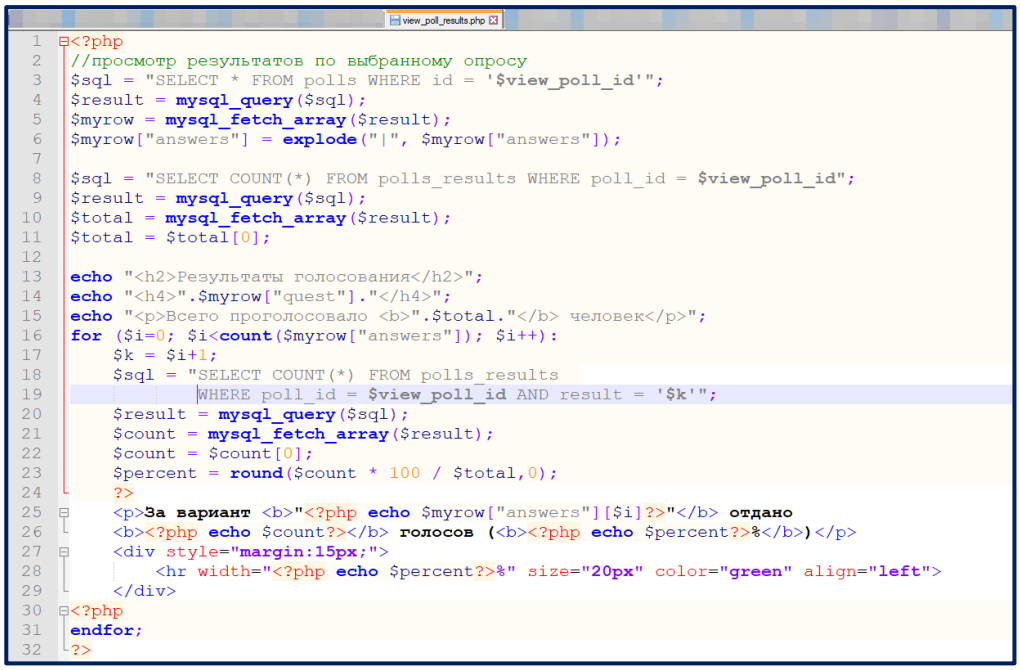


Рис. 130. Содержимое файла view_poll.php

5. Код файла **view_poll_results.php** выбирает из таблицы polls нужный опрос, затем подсчитывает из таблицы **polls_results** количество строк, для которых есть совпадения по **poll_id** (выбранный опрос). Это будет общее количество проголосовавших на данный опрос – **$total**. Затем в цикле нужно для каждого варианта подсчитать проголосовавших. **$percent** хранит процент проголосовавших. С помощью этой переменной **$percent** будем определять ширину тега **<hr>**, чтобы получилась мини-диаграмма.

6. Осталось разработать алгоритм, с помощью которого будем проверять, данный пользователь отвечал на выбранный опрос или нет. Механизм проверки будет основан на cookie, которые будут хранить список poll_id для данного пользователя. Если пользователь ранее не голосовал, то его результаты будут занесены в cookie (время хранения 1 час) и в базу данных. После голосования пользователя перенаправим на просмотр результатов. Также здесь определяется переменная $str_voted для вывода предупреждения.



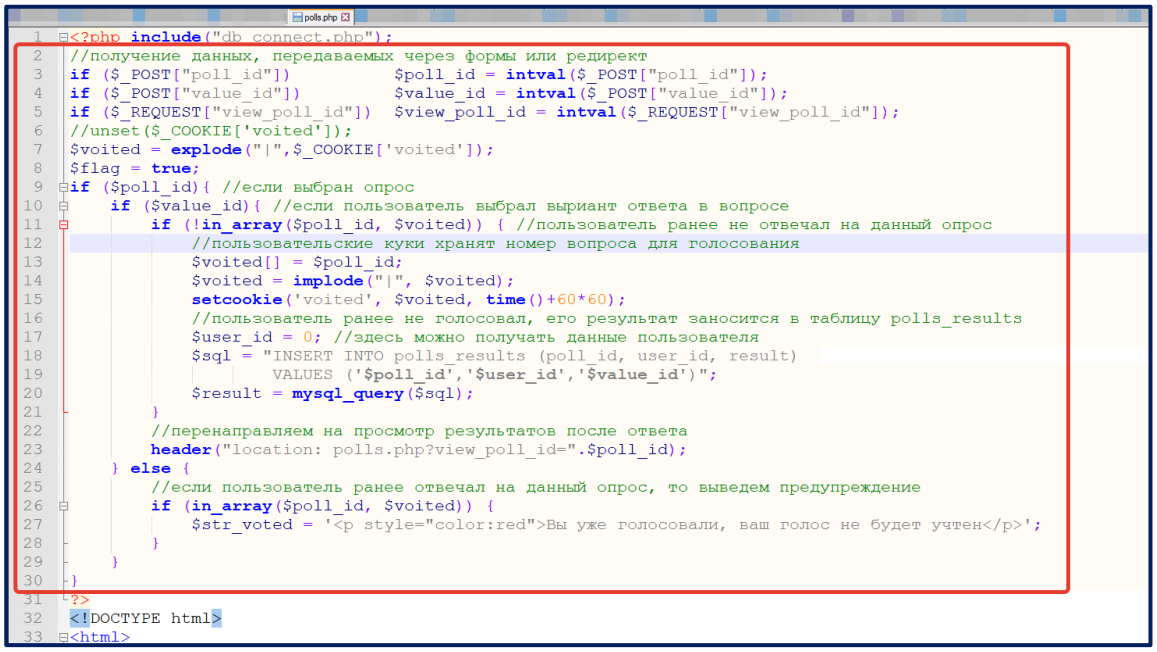
```php
<?php
//просмотр результатов по выбранному опросу
$sql = "SELECT * FROM polls WHERE id = '$view_poll_id'";
$result = mysql_query($sql);
$myrow = mysql_fetch_array($result);
$myrow["answers"] = explode("|", $myrow["answers"]);

$sql = "SELECT COUNT(*) FROM polls_results WHERE poll_id = $view_poll_id";
$result = mysql_query($sql);
$total = mysql_fetch_array($result);
$total = $total[0];

echo "<h2>Результаты голосования</h2>";
echo "<h4>".$myrow["quest"]."</h4>";
echo "<p>Всего проголосовало <b>".$total."</b> человек</p>";
for ($i=0; $i<count($myrow["answers"]); $i++):
    $k = $i+1;
    $sql = "SELECT COUNT(*) FROM polls_results
            WHERE poll_id = $view_poll_id AND result = '$k'";
    $result = mysql_query($sql);
    $count = mysql_fetch_array($result);
    $count = $count[0];
    $percent = round($count * 100 / $total,0);
    ?>
    <p>За вариант <b>"<?php echo $myrow["answers"][$i]?>"</b> отдано
    <b><?php echo $count?></b> голосов (<b><?php echo $percent?>%</b>)</p>
    <div style="margin:15px;">
        <hr width="<?php echo $percent?>%" size="20px" color="green" align="left">
    </div>
<?php
endfor;
?>
```

Рис. 140. Содержимое файла view_poll_results.php

Это, пожалуй, самая сложная часть алгоритма голосования, поэтому нужно внимательно прочитать комментарии в коде. Обратите внимание, что данный код вставляется в самое начало файла **polls.php**, но после подключения к базе данных, т.к. работа с cookie невозможна, если хоть какой-то символ html-кода уже был отправлен с сервера (рисунок 141).



```php
<?php include("db_connect.php");
//получение данных, передаваемых через формы или редирект
if ($_POST["poll_id"])          $poll_id = intval($_POST["poll_id"]);
if ($_POST["value_id"])         $value_id = intval($_POST["value_id"]);
if ($_REQUEST["view_poll_id"])  $view_poll_id = intval($_REQUEST["view_poll_id"]);
//unset($_COOKIE['voited']);
$voited = explode("|",$_COOKIE['voited']);
$flag = true;
if ($poll_id){ //если выбран опрос
    if ($value_id){ //если пользователь выбрал вырианта ответа в вопросе
        if (!in_array($poll_id, $voited)) { //пользователь ранее не отвечал на данный опрос
            //пользовательские куки хранят номер вопроса для голосования
            $voited[] = $poll_id;
            $voited = implode("|", $voited);
            setcookie('voited', $voited, time()+60*60);
            //пользователь ранее не голосовал, его результат заносится в таблицу polls_results
            $user_id = 0; //здесь можно получать данные пользователя
            $sql = "INSERT INTO polls_results (poll_id, user_id, result)
                    VALUES ('$poll_id','$user_id','$value_id')";
            $result = mysql_query($sql);
        }
        //перенаправляем на просмотр результатов после ответа
        header("location: polls.php?view_poll_id=".$poll_id);
    } else {
        //если пользователь ранее отвечал на данный опрос, то выведем предупреждение
        if (in_array($poll_id, $voited)) {
            $str_voted = '<p style="color:red">Вы уже голосовали, ваш голос не будет учтен</p>';
        }
    }
}
?>
<!DOCTYPE html>
<html>
```

Рис. 141. Код начала файла polls.php

7. Проверьте алгоритм: a) для пользователя, который голосует впервые, b) для повторного голосования. Покажите работу преподавателю.

4.4. Индивидуальные задания

Индивидуальные задания являются продолжением проектной методики освоения веб-программирования. Они выполняются студентом самостоятельно, но при необходимости преподаватель может оказать консультационную поддержку. Индивидуальные задания сформулированы в виде отдельных модулей разрабатываемого проекта, что позволяет студенту интегрировать знания из различных учебных дисциплин, сформировать умения в проектировании программного продукта [14], получить опыт практической разработки, а преподавателю – оценить знания, умения и навыки студентов в комплексе [10]. Также такая работа может лечь в основу портфолио студента, что для будущего IT-специалиста является особенно актуальным.

Каждое задание основано на ранее выполненных лабораторных работах, но тем не менее предполагает самостоятельный поиск подобных решений, их качественную адаптацию под свою задачу. В качестве помощи можно использовать учебные пособия как по языку программирования PHP [3; 5; 9; 18; 19; 20], так и по системе управления базами данных MySQL [2; 4; 6; 7; 9; 17; 20]. Студент выполняет одно задание, в некоторых случаях можно работать в паре над одним заданием.

Задания для индивидуальной работы

1. Спроектируйте базу данных для хранения фотографий на сайте. Во фронтальной части проекта должен быть выведен список опубликованных фотографий (можно сделать в виде красивой галереи или слайдера JavaScript). В бэкенде (панели администратора) проекта должна быть возможность просматривать список всех фото с небольшими эскизами фотографий и редактировать фото (загружать новое, изменять название). Также необходимо реализовать механизм публикации фото для фронтэнда проекта (сайта), удаление фотографии (и в БД и физически с сервера) и пересортировку.

2. Спроектировать базу данных для хранения комментариев на сайте.

Реализовать алгоритм комментирования статьи на сайте. Во фронтальной части проекта должен выводиться полный список постов, отсортированный по дате в обратном порядке. Выводиться должны только те сообщения, которые были опубликованы администратором (т.е. в отзывах присутствует модерация). С левой стороны должна находиться форма авторизации пользователей. Если авторизацию прошел пользователь с именем admin, то он может опубликовывать сообщения, удалять, а

также редактировать. Все остальные пользователи, в т.ч. и неавторизованные, могут оставлять сообщения на сайте.

Также в проекте должна быть реализована система оценивания прочитанного материала по 5-балльной шкале и вывод средней оценки всех опубликованных комментариев.

3. Спроектируйте базу данных для хранения информации о пользователях на сайте. Разработайте алгоритм формы для регистрации пользователей. Форма должна включать проверку правильности заполнения полей (имя, ник, e-mail, пароль, подтверждение пароля). Если пользователь вводит логин, который уже зарегистрирован, то должна выдаваться информация об ошибке сразу же (без отправки данных с формы обычным образом). То же самое, если пароль и подтверждение пароля не совпадут. Далее должна происходить стандартная регистрация с оповещением по e-mail (подтверждение регистрации по ссылке делать необязательно). Также должна присутствовать ссылка на восстановление пароля (восстановление по хешу, присланному по почте пользователя).

4. Спроектируйте на сайте базу данных для хранения отзывов на сайте. Реализуйте алгоритм, по которому зарегистрированный пользователь может оставить отзыв о сайте и поставить оценку по 5-балльной шкале. Отзывы не модерируются, а публикуются «как есть». При отправке отзыва владельцу сайта (администратору сайта) должно приходить письмо с текстом отзыва. На главной странице сайта должны размещаться три случайных отзыва и ссылка «Посмотреть все отзывы» с переходом на страницу всех отзывов. Самые последние отзывы должны выводиться в начале списка.

5. Спроектируйте базу данных для хранения опросов на сайте. Разработайте алгоритм, по которому пользователь может выбрать опрос из списка, ответить на этот опрос и получить статистическую информацию об этом опросе (сколько человек прошли опрос, как распределились ответы на опрос). Разработайте админ-панель для опросов, с помощью которой можно редактировать, публиковать опросы, а также просматривать статистику по опросам.

6. Спроектируйте базу данных для хранения цитат на сайте. Разработайте алгоритм, с помощью которого на вашем ресурсе будет выводиться случайная цитата ученого, философа или писателя. Предусмотрите вывод цитаты в виде тега

<blockquote>, фамилию ученого и год, когда эта фраза была озвучена. Разработайте админ-панель для редактирования, удаления и публикации цитат ученых.

7. Спроектируйте и реализуйте алгоритм тестирования по всем имеющимся в базе данных контрольным вопросам. В выполненных лабораторных работах контрольные вопросы выводятся для каждой статьи. Добавьте возможность пройти тест по всем имеющимся вопросам, причем для каждого вопроса должна указываться тема статьи/учебного материала, на проверку которого он направлен. Продумайте и реализуйте админ-панель для добавления, редактирования и публикации вопросов, а также прикрепления их к нужной статье/учебному материалу.

8. Спроектируйте базу данных для хранения новостей на сайте. Реализуйте алгоритм, выводящий три последних новости и ссылку «Все новости». При переходе по ссылке «Все новости» должна быть возможность посмотреть полный список опубликованных новостей. При этом должны быть доступны параметры сортировки: сортировка по возрастанию, сортировка по убыванию. Разработайте админ-панель для редактирования, добавления, удаления и публикации новостей.

9. Спроектируйте базу данных для хранения прикрепленных материалов. К каждой статье можно прикреплять неограниченное количество материалов: архивов, презентаций, текстовых документов и др. Все ссылки должны открываться в новом окне. Разработайте удобную админ-панель для загрузки прикрепленных материалов на сервер и прикрепления их к статьям/учебным материалам.

10. Разработайте поисковую систему на сайте. Поиск должен осуществляться по имеющимся таблицам в базе данных: поиск по статьям, контрольным вопросам, гостевой книге и пр. При выводе результатов должен показываться поисковый запрос, количество найденных результатов и список результатов с информацией, в какой таблице был найден материал (статьи, контрольные вопросы, гостевая книга и пр.).

11. Разработайте блок на сайте для вывода статистической информации. Это может быть количество зарегистрированных пользователей, самый популярный опрос, пользователь с самыми высокими баллами за тестирование, средний балл за тестирование по текущей статье/учебному материалу. В статистике использовать все виды агрегирующих функций: сумма, среднее, минимальное, максимальное, количество.

12. Спроектируйте специальный раздел для авторизованного пользователя – «Личный кабинет пользователя». В нем должна содержаться информация обо всех

пройденных тестах, количестве набранных баллов, сравнение со средним значением по данному тесту. Также в личном кабинете пользователя должна содержаться информация о его сообщениях в гостевой книге.

13. Спроектируйте базу данных для хранения часто задаваемых вопросов на сайте. Разработайте алгоритм, по которому выводятся часто задаваемые вопросы на сайте и ответы на них. Разработайте форму, через которую пользователь может отправить свой вопрос. После отправки вопроса программа должна отправлять два сообщения для пользователя и админа о том, что вопрос получен. Разработайте админ-панель для создания ответа на вопрос и публикации его на сайте.

14. Спроектируйте базу данных для хранения дополнительной литературы на сайте. Разработайте алгоритм, согласно которому, на странице со статьёй будет выводиться необходимая литература. Для литературы должны выводиться: автор, название, издательство, год издания, количество страниц. Разработайте админ-панель для редактирования списка литературы к статье/учебному материалу.

15. Спроектируйте базу данных для хранения писем рассылки. Разработайте алгоритм, который отправляет на e-mail всех пользователей письма с приглашением на ваш ресурс (изучить новый материал, пройти тестирование, прочитать новые сообщения гостевой книги и пр.). В скрипте с помощью специальных тегов отметьте имя (ник) пользователя. Пример: «Уважаемый(ая) Елена!» Шаблон текста сообщения должен сохраняться в базе данных. Его можно использовать многократно.

4.5. Задания для самостоятельной работы

1. Изучите приказ Федеральной службы по надзору в сфере образования и науки РФ от 14 августа 2020 г. № 831 «Об утверждении Требований к структуре официального сайта образовательной организации в информационно-телекоммуникационной сети "Интернет" и формату представления информации» [15].

Сформулируйте функционал отдельных модулей системы, которые можно реализовать на PHP и MySQL для сайта образовательной организации.

2. Изучите сервисы образовательных веб-ресурсов. Приведите примеры сервисов на обучающих веб-ресурсах. Найдите в сети Интернет и опишите сервисы обучающих веб-ресурсов. Изучите требования к образовательным сайтам. Разработайте структуру сайта образовательного учреждения.

3. Познакомьтесь с языком программирования PHP и системой управления базами данных MySQL. Изучите особенности языка программирования PHP и системы управления базами данных MySQL. Найдите и законспектируйте функции для работы с базой данных MySQL, графические функции в PHP.

4. Модифицируйте алгоритм тестирования лабораторной работы, добавьте возможность прохождения теста по таймеру.

5. Разработайте отдельный модуль на сайте: классный журнал, личный кабинет пользователя с домашними заданиями.

6. Изучите и законспектируйте функции, объекты и методы JavaScript. Разработайте гипертекстовый ресурс по функциям, объектам, методам JavaScript.

7. Измените алгоритм AJAX поиска, предложенный на лабораторном занятии, так, чтобы он работал не по нажатию кнопки OK, а по клавишам на клавиатуры (onkeypress, onkeyup, onkeydown). Реализовать на AJAX верификацию логина и пароля (проверка наличия логина в БД без перезагрузки страницы).

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Ansgar Becker, домашняя страница HeidiSQL. – URL: https://www.heidisql.com. – (дата обращения 31.08.2021).

2. Документация по MySQL. – URL: http://www.mysql.ru/docs/man/. – (дата обращения 04.07.2021).

3. Кисленко, Н.П. Интернет-программирование на PHP: учебное пособие / Н.П. Кисленко. – Новосибирск: Новосибирский государственный архитектурно-строительный университет (Сибстрин), ЭБС АСВ, 2015. – 177 с. – ISBN 978-5-7795-0745-5. – Текст: электронный // Электронно-библиотечная система IPR BOOKS: [сайт]. – URL: https://www.iprbookshop.ru/68769.html (дата обращения: 20.07.2021). – Режим доступа: для авторизир. пользователей. - DOI: https://doi.org/10.23682/68769.

4. Коды ошибок MySQL. – URL: http://allerrorcodes.ru/mysql/. – (дата обращения 31.08.2021).

5. Курс на портале intuit.ru «Web-программирование на PHP 5.2». – URL: http://www.intuit.ru/studies/courses/985/308/info. – (дата обращения: 31.08.2021).

6. Курс на портале intuit.ru «Введение в СУБД MySQL». – URL: http://www.intuit.ru/studies/courses/111/111/info. – (дата обращения: 31.08.2021).

7. Лебедева, Т.Н. Информационные системы и базы знаний: учебно-методическое пособие / Т.Н. Лебедева, Л.С. Носова, А.А. Рузаков. – Челябинск: Южно-Уральский государственный гуманитарно-педагогический университет, 2017. – 200 с. – ISBN 978-5-9069-0860-5.

8. Лебедева, Т.Н. Организация самостоятельной работы обучаемых при изучении программирования / Т.Н. Лебедева, Л.С. Носова // Методика преподавания математических и естественнонаучных дисциплин: современные проблемы и тенденции развития: Материалы V Всероссийской научно-практической конференции, Омск, 03 июля 2018 года / отв. ред. А.А. Романова. – Омск: Омский государственный университет им. Ф.М. Достоевского, 2018. – С. 226–229.

9. Мазуркевич, А. PHP. Настольная книга программиста: учебник / А. Мазуркевич, Д. Еловой. – Москва: Новое издание. – 2003. – 480 с.

10. Паршукова, Н.Б. Методика оценивания уровня сформированности икт-компетентности у будущих учителей информатики / Н.Б. Паршукова // Вестник Челя-

бинского государственного педагогического университета. – 2018. – № 4. – С. 74–86. – DOI 10.25588/CSPU.2018.69..4..008.

11. Паршукова, Н.Б. Проектирование и разработка образовательного портала / Н.Б. Паршукова. – Челябинск: Южно-Уральский государственный гуманитарно-педагогический университет, 2020. – 129 с. – ISBN 978-5-907409-118.

12. Паршукова, Н.Б. Работа над веб-проектом как способ совершенствования «гибких» навыков (soft skills) it-специалиста / Н.Б. Паршукова // Информатизация образования: проблемы и перспективы: Материалы V Международной научно-практической интернет-конференции, посвященной памяти Д.Ш. Матроса, Челябинск, 16 апреля 2021 года. – Челябинск, 2021. – С. 100–108.

13. Паршукова, Н.Б. Реализация проектной методики при обучении студентов разработке образовательных порталов / Н.Б. Паршукова // Информатизация образования: проблемы и перспективы: II Всероссийская научно-практическая конференция с международным участием, Челябинск, 27–28 марта 2014 года. – Челябинск: Челябинский государственный педагогический университет, 2014. – С. 83–89.

14. Паршукова, Н.Б. Формирование компетенции в области проектно-конструкторской деятельности при обучении ИТ-специалистов основам web-программирования / Н.Б. Паршукова // Междисциплинарный диалог: современные тенденции в гуманитарных, естественных и технических науках: сборник трудов IV Всероссийской научно-практической конференции преподавателей, ученых, специалистов и аспирантов, Челябинск, 17 марта 2015 года. – Челябинск: Полиграф-Мастер, 2015. – С. 235–242.

15. Приказ Федеральной службы по надзору в сфере образования и науки РФ № 831 от 14.08.2020 «Об утверждении Требований к структуре официального сайта образовательной организации в информационно-телекоммуникационной сети "Интернет" и формату представления информации». – URL: https://base.garant.ru/74901486/. – (дата обращения: 31.08.2021).

16. Рузаков, А.А. Особенности методики преподавания дисциплины «Управление данными» для направления подготовки бакалавриата «Информационные системы и технологии» профиля «Информационные технологии в образовании» / А.А. Рузаков // Современные технологии в физико-математическом образовании: сборник трудов II научно-практической конференции, Челябинск, 25–27 июня 2015 года / под. ред. С.А. Загребиной. – Челябинск: ЮУрГУ, 2015. – С. 54–58.

17. Рузаков, А.А. Управление данными: учебное пособие / А.А. Рузаков; Министерство образования и науки Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования «Челябинский государственный педагогический университет». – Челябинск: Челябинский государственный педагогический университет, 2015. – 132 с. – ISBN 978-5-9067-7740-9.

18. Руководство по PHP. – URL: https://www.php.net/manual/ru/. – (дата обращения 31.08.2021).

19. Савельева, Н.В. Основы программирования на PHP: курс лекций: Учебное пособие для студентов вузов, обучающихся по специальностям в области информационных технологий / Н.В. Савельева. – Электрон. текстовые данные. – Москва: Интернет-Университет Информационных Технологий (ИНТУИТ), 2005. – 264 с. – Режим доступа: http://www.iprbookshop.ru/22429. – ЭБС «IPRbooks», по паролю.

20. Суэринг, С. PHP и MySQL. Библия программиста / С. Суэринг; Стив Суэринг, Тим Конверс, Джойс Парк; [пер. с англ. и ред. К.А. Птицына]. – 2-е изд. – Москва [и др.]: Диалектика, 2010. – 911 с. – ISBN 978-5-845-916-402.

ПРИЛОЖЕНИЕ

Тест по PHP и MySQL

Вопрос 1. Что выведет данный код:

```
<?php
 echo "1" + "2";
?>
```

1) 12
2) 3
3) 1+2

Вопрос 2. Как сделать редирект (автоматическое перенаправление), например, на google.ru, на PHP?

1) header("Location: http://google.ru");
2) document.location = "http://google.ru";
3) location.href = "http://google.ru";
4) header("Redirect: http://google.ru");

Вопрос 3. Что выведет данный скрипт: <?php

```
function myfunc() {
   static $id = 0;
   $id++;
   echo $id;
 }
 myfunc();
 myfunc();
 myfunc();
?>
```

1) 123
2) 111
3) 222
4) 333

Вопрос 4. Программист написал такой код:

```
<?php
 for ($i = 0; $i < 5; $i++) {
  if ($i % 2 == 0) continue;
  echo $i;
 }
?>
```

Что он увидит в результате?

1) 13
2) 013
3) 024
4) 24

Вопрос 5. Как задаются комментарии в языке PHP?

1) /* несколько строк комментариев */
2) / несколько строк комментариев /
3) ** строка комментариев **
4) <!-- несколько строк комментариев - >

Вопрос 6. Укажите числовые типы данных, которые имеются в языке PHP?

1) Float, integer
2) String
3) Text
4) Boolean, integer
5) Resource, float

Вопрос 7. Какая функция предназначена для того, чтобы закрыть соединение с сервером mysql? Какие параметры можно передать в эту функцию?

1) mysql_close. Параметры: указатель на ресурс, связанный с БД.
2) mysql_connect. Параметры: адрес сервера, имя пользователя, флаг нового соединения, пользовательские флаги.
3) mysql_query. Параметры: запрос, указатель на ресурс, связанный с БД.

Вопрос 8. В базе данных book в таблице persons хранится следующая информация о человеке: имя (поле first_name), фамилия (поле last_name), адрес электронной почты (поле email). Получить из этой базы данных информацию обо всех людях с фамилией «Иванов». Укажите правильный фрагмент кода для данной задачи.

1)

```php
<?php
    $db = mysql_connect("localhost", "admin", "123");
    mysql_select_db("book",$db);
    $result = mysql_query("SELECT * FROM persons WHERE last_name='Иванов'", $db);
    while ($myrow = mysql_fetch_array($result)) {
        echo "name: ".$myrow["first_name"]."<br/>";
        echo "email:".$myrow["email"]."<br/><br/>";
    }
?>
```

2)

```php
<?php
    $db = mysql_connect("localhost", "admin", "123");
    $result = mysql_query("SELECT * FROM persons WHERE last_name='Иванов'", $db);
    while ($myrow = mysql_fetch_array($result)) {
        echo "name: ".$myrow["first_name"]."<br/>";
        echo "email:".$myrow["email"]."<br/><br/>";
    }
?>
```

3)

```php
<?php
    $db = mysql_connect("localhost", "admin", "123");
    $result = mysql_query("SELECT * FROM book WHERE last_name='Иванов'", $db);
    while ($myrow = mysql_fetch_array($result)) {
        echo "name: ".$myrow["first_name"]."<br/>";
        echo "email:".$myrow["email"]."<br/><br/>";
    }
?>
```

4)

```php
<?php
    $db = mysql_connect("localhost", "admin", "123");
    $result = mysql_query("SELECT * FROM book.persons WHERE name='Иванов'", $db);
    while ($myrow = mysql_fetch_array($result)) {
        echo "name: ".$myrow["first_name"]."<br/>";
        echo "email:".$myrow["email"]."<br/><br/>";
    }
?>
```

Вопрос 9. Дана команда:

if ($var) echo "Hello"; else echo "Bye";

В каком случае на экран будет выведено слово «Bye»?

1) Если $var == false
2) Если $var == true
3)

Вопрос 10. Чем отличается оператор **break** от оператора **continue**?

1) **break** заканчивает выполнение текущего цикла, а **continue** – текущей итерации цикла
2) **break** используется для принудительной остановки циклов, а **continue** – для условных операторов

3) **continue** не может быть вызван с дополнительным числовым аргументом, а **break** – может

Вопрос 11.

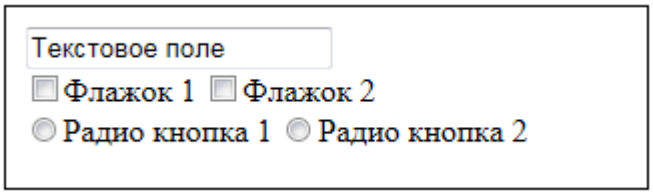
С помощью какой функции можно удалить файл?

1) Unlink()
2) Fclose()
3) File_delete()

Вопрос 12. Каким образом можно уничтожить переменную сессии?

1) session_destroy()
2) Функция session_unregister(имя_переменной) удаляет глобальную переменную из текущей сессии (т.е. удаляет ее из списка зарегистрированных переменных)

Вопрос 13. Какие тэги были использованы при создании формы ниже?



1) <input type='text'>, <input type='checkbox'> и <input type='radio'>
2) <input type='textfield'>, <input type='checkbox'> и <input type='radiobutton'>
3) <input type='text'> и <input type='check'>
4) <input type='text'>, <input type='check'> и <input type='radio'>

Вопрос 14. Укажите тег HTML, позволяющий определить поле для ввода пароля.

1) <input type='password' />
2) <select type='password' />
3) <input name='password' />
4) <input type='text' name='password' />
5) <password>

Вопрос 15. Укажите запрос MySQL, который выбирает одну запись из таблицы guest и удаляет ее. Учесть, что переменная $id определена и содержит числовое значение.

1) DELETE FROM guest WHERE id = '$id'
2) SELECT * FROM guest WHERE id = '$id'
3) DELETE FROM guest
4) DELETE * FROM guest WHERE id = '$id'

Вопрос 16. Таблица articles содержит поля id, title (название), description (описание), text, author («автор»), data (в формате YYYY-MM-DD, т.е. 2011-11-04 – 4 ноября 2011 г.).

Укажите запрос MySQL, который выбирает из таблицы articles все записи, поле «автор» которых Петров, и записи созданы после 1.02.2012.

1) SELECT * FROM articles WHERE data > ‘2012-02-01’ AND author = ‘Петров’
2) SELECT * FROM articles WHERE id > ‘1.02.2012’ AND author = ‘Петров’
3) SELECT id, author, * FROM articles WHERE id > ‘1.02.2012’ AND author = ‘Петров’
4) SELECT * FROM articles WHERE data > ‘2012-01-02’ AND author = ‘Петров’

Вопрос 17. Укажите условные конструкции языка программирования PHP.

1) If, if … else …, switch
2) If, if … then … else, case
3) If, foreach
4) If, do, while

Вопрос 18. Укажите правильный формат оператора выбора в PHP.

1) switch (выражение или переменная){
   case значение1: блок_действий1; break;
   case значение2: блок_действий2; break;
   default: блок_действий_по_умолчанию; break;
   }
2) case (выражение или переменная){
   switch значение1: блок_действий1; break;
   switch значение2: блок_действий2; break;
   default: блок_действий_по_умолчанию;
   }
3) switch (выражение или переменная){
   case значение1: блок_действий1;
   case значение2: блок_действий2;
   default: блок_действий_по_умолчанию
   }
4) switch (выражение или переменная){
   case значение1: блок_действий1; die();
   case значение2: блок_действий2; die();
   default: блок_действий_по_умолчанию; die();
   }

Вопрос 19. Выберите НЕВЕРНОЕ описание цикла с предусловием.

1) while (выражение): блок_выполнения end;
2) while (выражение) { блок_выполнения }

3) while (выражение): блок_выполнения endwhile;
4) while (выражение) оператор;

Вопрос 20. Выберите ВЕРНОЕ описание цикла с параметром.

1) for (начальное_выражение; условие; выражение_управления_циклом)
   {
   // последовательность операторов
   }
2) foreach (начальное_выражение; условие; выражение_управления_циклом)
   {
   // последовательность операторов
   }
3) while (начальное_выражение; условие; выражение_управления_циклом)
   {
   // последовательность операторов
   }

Вопрос 21. Есть ли разница между одинарными и двойными кавычками при интерпретации значения литералов строкового типа?

1) Да
2) Нет

Вопрос 22. Что напечатает следующий код:

```
$i = 0;
do {
    $i++;
} while ($i > 10);
echo $i;
```

1) 1
2) 10
3) 9
4) 0

Вопрос 23. Необходимо перенаправить пользователя средствами PHP на другую страницу, но при тестировании данного кода Вы заметили, что при работе программы возникает ошибка.

```
<?php
        echo “Сейчас вы будете перенаправлены на другую страницу”;
        header(“Location: http://example.com”);
```

?>

1) Функцию header() необходимо вызывать до отправки любого вывода
2) Функции header() не существует. Необходимо использовать функцию headers_list()
3) Функции header() имеет второй обязательный параметр, который необходимо указать в данном фрагменте кода
4) Функции echo не существует. Необходимо использовать другую функцию
5) Функции header() не существует. Необходимо использовать функцию headers_sent()

Вопрос 24. Какие массивы называют ассоциативными?

1) те, у которых в качестве индексов используются строки
2) те, у которых в качестве индексов используются дробные числа
3) те, у которых в качестве индексов используются идентификаторы переменных
4) те, у которых в качестве индексов используются целые числа

Вопрос 25. Сколько разных типов циклов есть в PHP?

1) 4
2) 3
3) 2
4) 1

Вопрос 26. Какая функция PHP используется для удаления файла?

1) unlink()
2) remove()
3) del()
4) move()
5) unset()

Вопрос 27. Дана форма

```
<form action="script.php" method="post">
    <label>Input:</label>
    <input type="text" name="field" />
</form>
```

В каком массиве будут храниться данные после сабмита формы на сервер?

1) $_POST[]
2) $_GET[]
3) $_FORM[]
4) $_FORMS[]
5) $POST[]

6) $GET[]

Вопрос 28. Где обычно выполняются скрипты, написанные на PHP?

1) На сервере
2) Внутри операционной системы
3) На клиенте
4) В окне браузера

Вопрос 29.

Как правильно вызвать функцию func с одним параметром?

1) func(2)
2) invoke func(3)
3) func(param=4)
4) call func(1)

Вопрос 30. Для чего предназначена функция isset()?

1) Она проверяет, была ли инициализирована переменная
2) Она проверяет, существует ли массив
3) Она проверяет, существует ли объект
4) Ничего из вышеперечисленного

Вопрос 31. Укажите оператор включения кода из другого файла

1) include
2) $include
3) $isset
4) Isset
5) require_one

Вопрос 32. Соотнесите оператор включения кода из другого файла с его особенностями.

1) include
2) include_once
3) require
4) require_once

A. позволяет включать код, содержащийся в указанном файле, и выполнять его столько раз, сколько программа встречает этот оператор
B. если код из файла уже был включен, то его нельзя
C. включить повторно

D. позволяет включать в программу и исполнять какой-либо файл, при этом, если подключить файл не удалось, выдается фатальная ошибка, прекращающая дальнейшие действия алгоритма

E. позволяет включать в программу и исполнять какой-либо файл только один раз, при этом, если подключить файл не удалось, выдается фатальная ошибка, прекращающая дальнейшие действия алгоритма.

Учебное издание

Наталья Борисовна Паршукова

ПРОГРАММИРОВАНИЕ С ИСПОЛЬЗОВАНИЕМ PHP И MYSQL В РАЗРАБОТКЕ ВЕБ-ПРИЛОЖЕНИЙ

УЧЕБНОЕ ПОСОБИЕ

ISBN 978-5-907409-81-1

Работа рекомендована РИС университета
Протокол № 24 от 2021г.

Издательство ЮУрГГПУ
454080, г. Челябинск, пр. Ленина, 69

Редактор О.В. Угрюмова
Технический редактор О.В. Угрюмова

Подписано в печать 29.10.2021
Формат 60х84/8 Объем 5,3 уч.-изд. л (18,9 усл. п. л.)
Тираж 100 экз. Заказ №

Отпечатано с готового оригинал-макета в типографии ЮУрГГПУ

454080, г. Челябинск, пр. Ленина, 69