МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
ЛУГАНСКОЙ НАРОДНОЙ РЕСПУБЛИКИ
ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
ЛУГАНСКОЙ НАРОДНОЙ РЕСПУБЛИКИ
«ЛУГАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ ВЛАДИМИРА ДАЛЯ»

Стахановский инженерно-педагогический институт менеджмента
Кафедра информационных систем

МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ВЫПОЛНЕНИЮ ЛАБОРАТОРНЫХ РАБОТ

по дисциплине

«КОМПЬЮТЕРНЫЕ И ТЕЛЕКОММУНИКАЦИОННЫЕ ТЕХНОЛОГИИ В ПРОФЕССИОНАЛЬНОЙ ДЕЯТЕЛЬНОСТИ»

для студентов направления подготовки
Профессиональное обучение (по отраслям),
магистерская программа
«Информационные технологии и системы» (в 2 х частях). Часть 2

Луганск 2023

УДК 69.032/007, 612.313

Рекомендовано к изданию Учебно-методическим советом
ГОУ ВО ЛНР «ЛГУ им. В. ДАЛЯ»
(протокол № от 2023 г.)

Методические указания к выполнению лабораторных работ по дисциплине **«Компьютерные и телекоммуникационные технологии в профессиональной деятельности»** для студентов направления подготовки **Профессиональное обучение (по отраслям),** магистерская программа «Информационные технологии и системы» в 2-х частях. Часть 2 / Сост.: Д. С. Тимошенко. – **Стаханов**: ГОУ ВО ЛНР «ЛГУ им. В. ДАЛЯ», 2023. – 71с.

Методические указания к выполнению лабораторных работ содержит двенадцать работ по дисциплине «Компьютерные и телекоммуникационные технологии в профессиональной деятельности», которые включают в себя теоретические сведения, примеры решения, задачи и варианты с данными для их выполнения. К каждой лабораторной работе приведены вопросы для самопроверки. В методических рекомендациях представлен список использованных источников.

Предназначены для студентов магистерской программы «Информационные технологии и системы».

Составители: ст. преп. Тимошенко Д.С.

Ответственный за выпуск: доц. Карчевский В.П.

Рецензент: доц. Черникова С.А.

СОДЕРЖАНИЕ

Аннотация

Для закрепления теоретических знаний и приобретения необходимых умений, программой учебной дисциплины предусмотрено проведение лабораторных занятий. Для выполнения лабораторных занятий составлены методические указания, которые содержат краткий теоретический материал и практические примеры создания веб-страниц.

Целями освоения дисциплины Компьютерные сети и телекоммуникации является обучение теоретическим и практическим основам в организации и функционировании компьютерных сетей; формирование у студентов понимания важности применения и развития вычислительных систем, сетей и телекоммуникаций в современных технологиях, а также обучение студентов общим принципам построения вычислительных систем различных архитектур, принципам организации и характеристикам составных элементов компьютерных сетей, принципам и технологиям организации систем передачи данных.

Лабораторная работа №5

Тема:PHP. Создание скриптов

Цель:Научиться создавать элементарные сценарии на языкеPHP

Задание для выполнения

Создайте любую web-страницу. Для нее

1. Добавьте календарь в текущий месяц с названием текущего месяца и номером года. Выделите текущую дату.

2. Добавьте генератор паролей.

3. Добавьте форму для отправки сообщений по электронному адресу.

4. Найдите, изучите принцип работы и добавьте на ваш сайт любой интересный скрипт, созданный с помощью PHP (например, несложную игру или визуальные эффекты).

Отчет должен содержать:

1. Тему, цель работы
2. Коды созданных файлов и копию окна обозревателя при их просмотре.

Методические указания

Календарь

Использование функции date().Функция date() принимает два аргумента, один из которых является необязательным. Первый аргумент представляет собой строку формата, а второй, необязательный, – метку времени UNIX. Если метка времени не указана, функция date() обрабатывает текущую дату и время. Она возвращает отформатированную строку и содержит дату. Типичный вызов функции выглядит так:

```
echo date("jS F Y");
```

Вывод этого выражения имеет вид "31th July 2001". Коды форматирования, используемые функцией date(), перечислены в табл. 5.1.

Таблица 5.1 – Коды форматирования функции PHP date()

Код	***Описание***
a	Утро или время после полудня, с двумя строчными символами, "am" или "pm"
A	Утро или время после полудня, с двумя прописными символами, "AM" или "PM".
B	Internet-время Swatch – универсальная часовая схема. Детальнее о ней можно узнать на сайте http://www.swatch.com.
d	День месяца в виде двузначного числа с ведущим нулем. Диапазон значений – от "01" до "31".
D	День недели в виде трехбуквенной аббревиатуры. Диапазон значений – от Mon (понедельник) до Sun (воскресенье).
F	Луна в полнотекстовом формате. Диапазон значений – от January до December.
g	Часы в 12-часовом формате без ведущих нулей. Диапазон значений – от "1" до "12".
G	Часы в 24-часовом формате без ведущих нулей. Диапазон значений – от "0" до "23".
h	Часы в 12-часовом формате с ведущими нулями. Диапазон значений – от "01" до "12"
H	Часы в 24-часовом формате с ведущими нулями. Диапазон значений – от "00" до "23"
i	Минуты с ведущими нулями. Диапазон значений – от "00" до "59".
I	Переход на летнее время представлен значением логического типа. Если переход на летнее время установлен, функция возвращает значение "1", в противном случае - "0".
j	День месяца в виде числа без ведущих нулей. Диапазон значений – от "1" до "31".
I	День недели в полнотекстовом формате. Диапазон значений – от "Monday" (понедельник) до "Sunday" (воскресенье).
L	Высокосный год, представленный значением логического типа. Функция возвращает значение "1", если дата принадлежит високосному году, и "0" – иначе.
m	Луна в двузначном числовом формате с ведущими нулями. Диапазон значений – от "01" до "12".
M	Луна в виде трехбуквенной аббревиатуры. Диапазон значений – от Jan (январь) до Dec (де-

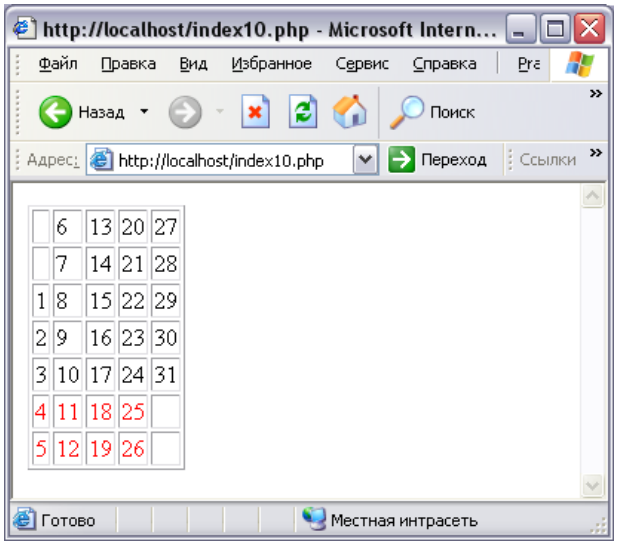
Код	***Описание***
	кабрь).
n	Луна в виде числа без ведущих нулей. Диапазон значений – от "1" до "12".
s	Секунды с ведущими нулями. Диапазон значений от "00" до "59".
S	Порядочный суффикс для дат в двухбуквенном формате. Он может принимать значения "st", "nd", "rd" или "th" в зависимости от числа, за которым он следует.
t	Полное количество дней в месяце. Диапазон значений – от "28" до "31".
T	Временная зона сервера, заданная в трехбуквенном формате, например, "EST".
U	Число секунд с 1 января 1970 г. по настоящее время; его также называют меткой времени UNIX для текущей даты.
w	День недели в виде номера. Диапазон значений – от "0" (воскресенье) до "6" (суббота).
B	Год в двузначном формате, например, "00".
Y	Год в четырехзначном формате, например, "2000".
z	День в виде числа. Диапазон значений – от "0" до "365".
Z	Смещение текущей временной зоны в секундах. Диапазон значений – от "-43200" до "43200".

Пример использования функции date() приведен ниже для создания календаря на текущий месяц.

Листинг PHP-кода страницы с календарем

```php
<?php
 // Вычисляем количество дней в данном месяце
 $dayofmonth = date('t');
 // Счетчик для дней месяца
 $day_count = 1;

 // 1. Первая неделя
 $num = 0;
 for($i = 0; $i < 7; $i++)
 {
  // Вычисляем номер для недели для числа
  $dayofweek = date('w', mktime(0, 0, 0, date('m'), $day_count, date('Y')));
  // Приводим числа к формату 1 – понедельник, ..., 6 – суббота
  $dayofweek = $dayofweek - 1;
  if($dayofweek == -1) $dayofweek = 6;

  if($dayofweek == $i)
  {
   // Если дни недели соапрадают,
   // заполняем массив $week
   // числами месяца
   $week[$num][$i] = $day_count;
$day_count++;
}
else
 {
  $week[$num][$i] = "";
 }
}

// 2 Следующие недели месяца
while(true)
{
 $num++;
 for($i = 0; $i < 7; $i++)
 {
  $week[$num][$i] = $day_count;
  $day_count++;
  // Если дошли до конца месяца - выходим
  // из цикла
  if($day_count > $dayofmonth) break;
 }
 // Если дошли до конца месяца - выходим
 // из цикла
 if($day_count > $dayofmonth) break;
}

// 3. Выводим седержмое массива $week
// в виде календаря
// Выводим таблицу
echo "<table border=1>";
for($j = 0; $j < 7; $j++)
{
 echo "<tr>";
 for($i = 0; $i < count($week); $i++)
 {
  if(!empty($week[$i][$j]))
  {
   // Если есть суббота и воскресенье
   // выделяем их
   if($j == 5 || $j == 6)
       echo "<td><font color=red>".$week[$i][$j]."</font></td>";
   else echo "<td>".$week[$i][$j]."</td>";
  }
  else echo "<td> </td>";
 }
 echo "</tr>";
}
echo "</table>";

?>
```

Результат исполнения скрипта



	6	13	20	27
	7	14	21	28
1	8	15	22	29
2	9	16	23	30
3	10	17	24	31
4	11	18	25	
5	12	19	26	

Рисунок 5.1 – Страница с календарем

Генератор паролей

Есть несколько вариантов создания скриптов генерации паролей. В первом случае пароль генерируется случайным образом посредством функцииuniqid. Эта функция возвращает уникальный идентификатор, основываясь на значении текущего времени в микросекундах.

При таком варианте использования функции возвращается 128-битный хэш-код

Генерация пароля по алгоритму MD5
$code = '282a586f2cd5e9d42297480304990d1e';
highlight_string($code);

Однако полученный пароль содержит только буквы английского языка в нижнем регистре и цифры. Для генерации более устойчивого к подбору паролей можно воспользоваться нижеприведенным скриптом.

Листинг PHP-кода страницы с генерацией равномерного пароля

```
<html>
<head>
<title> parol </title>
</head>
<body>

<form method=post>
<p> Введите длину пароля </p>
<input type=text name=number>
<input type=submit value="Генерировать">
<form><br><br>
<? php

 // Параметр $number – количество символов в пароле
 echo generate_password($_POST['number']);
 function generate_password($number)
```

```
{
  $arr = array('a','b','c','d','e','f',
               'g','h','i','j','k','l',
               'm','n','o','p','r','s',
               't','u','v','x','y','z',
               'A','B','C','D','E','F',
               'G','H','I','J','K','L',
               'M','N','O','P','R','S',
               'T','U','V','X','Y','Z',
               '1','2','3','4','5','6',
               '7','8','9','0','.',',',
               '(',')','[',']','!','?',
               '&','^','%','@','*','$',
               '<','>','/','|','+','-',
               '{','}','`','~');

  //Генерируем пароль

  $pass = "";
  for($i = 0; $i < $number; $i++)
  {
    // Находим случайный индекс массива
    $index = rand(0, count($arr) - 1);
    $pass .= $arr[$index];
  }
  return $pass;
}
?>
</body>
</html>
```

Далее приведена страница с формой для генерации паролей и сгенерированным паролем

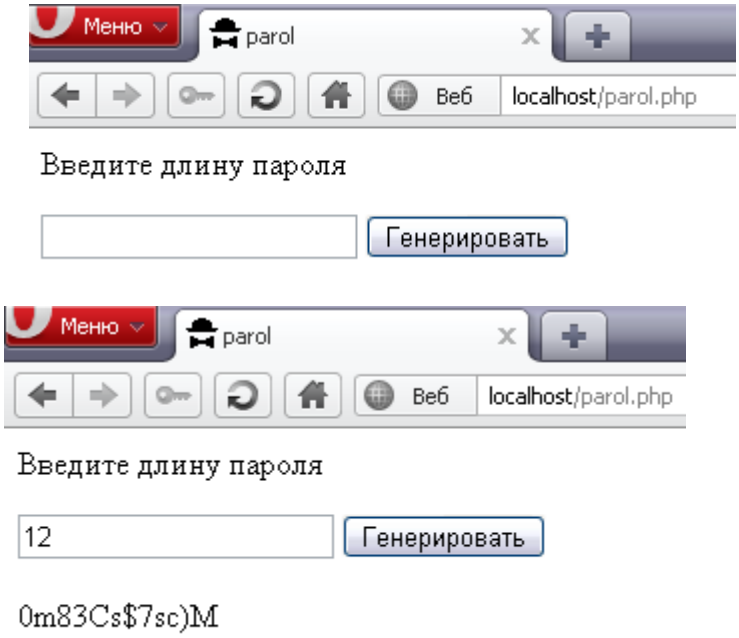


Рисунок 5.2 – Результат генерации пароля

Отправка сообщений с вложением на электронную почту

Функция предназначена для отправки письма с сайта с вложенным файлом и разработана участниками форума Trianon и elenaki при совместном обсуждении.

Создадим HTML-форму, предназначенную для заполнения пользователем

```
<HTML>
<HEAD>
<TITLE>Отправка сообщения с вложением</TITLE>
</HEAD>
<BODY>
<H3> <center><font color=#1E90FF>Отправка сообщения с
                    вложением</font></H3>
<center>
<table width=1 border=0>

<form action=mail.php enctype='multipart/form-data' method=post>
<tr><td width=50%>To:</td><td align=right><input type=text name=mail_to
        maxlength=32></td></tr>
<tr><td width=50%>Subject:</td><td align=right><input type=text
        name=mail_subject maxlength=64></td></tr>
<tr><td colspan=2>Сообщение:<br><textarea cols=50 rows=8
        name=mail_msg></textarea></td>
<tr><td width=50%>Photo:</td><td align=right><input type=file
        name=mail_file maxlength=64></td></tr>
</tr><tr><td colspan=2><input type=submit value='Отправить'></td></tr>
</form>
</table>
</center>
</BODY>
</HTML>
```

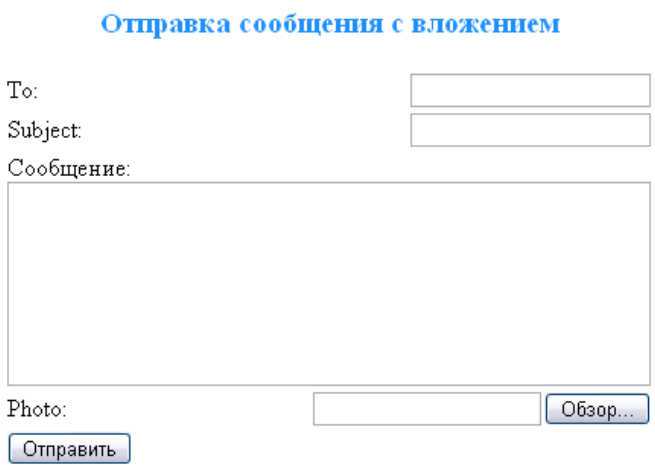


Рисунок 5.3 – Форма для отправки сообщения

Обработчик формы:

```
<?
if(empty($_POST['mail_to'])) exit("Введите адрес получателя");

// проверяем правильность заполнения с помощью регулярного выражения
if (!preg_match("/^[0-9a-z_] +@ [0-9a-z_^\.]+\.[az]{2,3}$/i", $_POST['mail_to'] )) ex-
    it("Введите адрес в виде
 somebody@server.comsomebody ");
$picture = "";

// Если поле выбора вложения не пустое - закачиваем его на сервер
if (!empty($_FILES['mail_file']['tmp_name']))
```

```
{
// Закачиваем файл
$path = $_FILES['mail_file']['name'];
if (copy($_FILES['mail_file']['tmp_name'], $path)) $picture = $path;
}

$thm = $_POST['mail_subject'];
$msg = $_POST['mail_msg'];
$mail_to = $_POST['mail_to'];

// Отправляем почтовое сообщение
if(empty($picture)) mail($mail_to, $thm, $msg);
else send_mail($mail_to, $thm, $msg, $picture);

// Вспомогательная функция для отправки почтового сообщения с вложением

function send_mail($mail_to, $thema, $html, $path)
{ if ($path) {
$fp = fopen($path, "rb");
if (!$fp)
{ print "Cannot open file";
exit();
}

$file = fread($fp, filesize($path));
fclose($fp);
}

$name = "file.ext"; // в этой переменной надо сформировать имя файла (без всякого
   пути)
$EOL = "\r\n"; // ограничитель строк, некоторые почтовые сервера требуют \n - по-
   добрать опытным путем
$boundary = "--".md5(uniqid(time())); // любая строка, которой не будет ниже в по-
   токе данных.

$headers = "MIME-Version: 1.0;$EOL";
$headers .= "Content-Type: multipart/mixed; boundary=\"$boundary\"$EOL";
$headers .= "From:
 address@server.comaddress ";
$multipart = "--$boundary$EOL";
$multipart .= "Content-Type: text/html; charset=windows-1251$EOL";
$multipart .= "Content-Transfer-Encoding: base64$EOL";
$multipart .= $EOL; // раздел между заголовками и телом html-части
$multipart .= chunk_split(base64_encode($html));
$multipart .= "$EOL--$boundary$EOL";
$multipart .= "Content-Type: application/octet-stream; name=\"$name\"$EOL";
```

```
$multipart .= "Content-Transfer-Encoding: base64$EOL";
$multipart .= "Content-Disposition: attachment; filename=\"$name\"$EOL";
$multipart .= $EOL; // раздел между заголовками и телом прикрепленного файла
$multipart .= chunk_split(base64_encode($file));
$multipart .= "$EOL--$boundary--$EOL";
if(!mail($mail_to, $thema, $multipart, $headers))
{return False; //если не письмо не отправлено
}
else { //// если письмо отправлено
return True;
}
exit;
}
?>
```

Контрольные вопросы

1. Для чего предназначена функция date()?
2. Какие параметры имеет функция date()?
3. Как часто используется эта функция?
4. Как можно генерировать пароли?
5. Как отправить письмо с вложением? Какие функции при этом используются?

Лабораторная работа №6

Тема:Взаимодействие PHP и MySQL

Цель:Научиться прорабатывать базы данных**MySQL**с помощью сценариев на языке PHP

Задание для выполнения

1. Ознакомиться с методическими указаниями по данной теме.
2. Создать согласно варианту однотабличную базу данных, содержащую 4 произвольных поля разных типов.

3. Заполнить таблицу 10 записями.
4. Отобразить данные из таблицы в окне обозревателя.
5. Выбрать данные из таблицы, удовлетворяющие любому условию (три разных запроса).
6. Измените данные для двух записей в таблице.
7. Выполнить сортировку данных по одному из полей.
8. Удалите несколько записей из таблицы.

Вариант	***Тема***
1	Автошкола
2	Автосалон
3	Библиотека
4	Аэропорт
5	Банк
6	Поликлиника
7	Строительные материалы
8	Аптека
9	Компьютерный клуб
10	Туристическое агентство
11	Косметический салон
12	Фотосалон
13	Железнодорожный вокзал
14	Автовокзал
15	Магазин кондитерских изделий
16	Институт
17	Налоговая инспекция
18	Морской порт
19	Музей
20	Школа

Отчет должен содержать:

1. Тему, цель работы

2. Коды созданных файлов и копию окна обозревателя при их просмотре.

Методические указания

Базы данных

В PHP включена обширная поддержка различных серверов баз данных, что позволяет создавать комплексные приложения, обрабатывающие значительные объемы данных. Все типы серверов баз данных, поддерживаемых PHP, приведены в табл. 6.1.

Таблица 6.1 – СУБД, поддерживаемые в СРН

Adabas	Front Base	Ingres	MySQL
PostgreSQL	Unix dbm	dBase	Hyperwave
InterBase	ODBC	Solid	Velocis
Direct	Empress	IBMDB2	mSQL
Oracle	SQLIte	FilePro	Informix
MS-SQL	Ovrimos	Sybase	

Что такое база данных? База данных является централизованным хранилищем, предназначенным для хранения, доступа и обработки данных под управлением прикладной программы.

Чаще базы данных строятся на основе таблиц. Но, в отличие от бумажного документа, таблица базы данных предоставляет гораздо больше возможностей для обработки информации, в частности, сортировки, поиска, добавления, изменения и удаления строк. Есть возможность связывать разные таблицы между собой путем создания отношений (англ. relations). Базы данных, допускающие создание отношений меж таблицами, именуются реляционными.

Каждый отдельный элемент данных в таблице представляет собой поле таблицы. Объединение полей представляет собой запись. Каждая запись в таблице представлена отдельной строкой, а каждый столбец в строке является полем. Набор строк, каждая из которых содержит один и тот же набор полей, образует таблицу базы данных.

Так что такое база данных? В простой форме база данных представляет собой набор из одной или нескольких таблиц. Для доступа к данным в таблицах обычно используется специальный язык. Один из таких языков – SQL, рассмотрим далее.

Настройка PHP для работы с базами данных

Настройка PHP для работы с базами данных может быть несколько утомительной и зависит от используемой операционной системы.

В Windows поддержка баз данных осуществляется с помощью библиотек расширения, которые размещаются в подкаталоге ext каталога, в который установлен PHP (обычно это C:\PHP\ext). Этот каталог задается в файле конфигурации php.ini в параметре extensiondir. Для подключения того или иного расширения следует добавить в конфигурационный файл php.ini строчку вида extension=name, где name – имя соответствующей библиотеки. Обычно все необходимые строчки уже присутствуют в файле

php.ini, но отключены с помощью символа комментария («;»). Поэтому достаточно удалить этот символ в соответствующей строке файла. Если PHP подключен как модуль Web-сервера, то после этого требуется перезапуск службы Web-сервера. Например, для подключения поддержки MYSQL следует убрать символ «;» в строке, подключающей библиотеку phpmysql.dll:

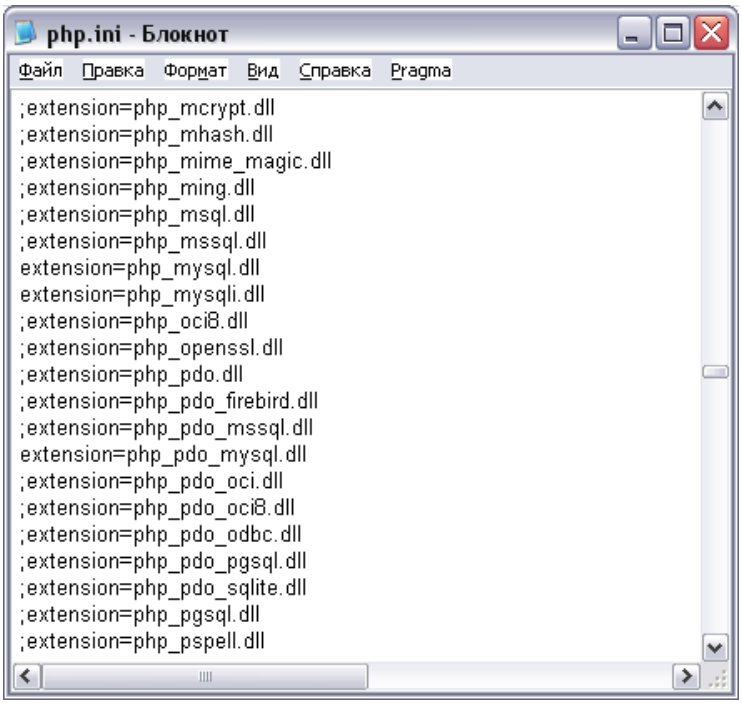


Рисунок 6.1 – Содержимое файла php.ini

Создание базы данных

Первым шагом при работе с SQL-сервером является создание базы данных. Для этого необходимо установить соединение с MSQL, осуществляемым с помощью функцииmysql_connect(при необходимости следует задать другие имя и/или пароль пользователя):

```
$connection = mysql_connect ("localhost", "user", "password") or die ("Ошибка соединения с сервером");
```

Далее с помощью функции mysql_query выполняется SQL-запрос, создающий базу данных.

```
$query = "CREATE DATABASE IF NOT EXISTS dt>";
$result = mysql_query ($query) or die("Ошибка при выполнении запроса:
".mysql_error());
```

Наконец, происходит выбор только что созданной базы данных с помощью функцииmysql_select_db. После завершения работы с базой данных необходимо разорвать соединение с SQL-сервером, для чего предназначена функциямysql_close. Порядок создания базы данных показан на следующем примере. После того, как база данных будет успешно создана, можно перейти к созданию таблиц.

```
<html>
<head>
<title> Создание базы данных </title>
</head>
<body>
<center>
<h1> Создание базы данных </h1>
<?
$connection = mysql_connect ("localhost", "root","") or die ("Ошибка
   соединения с сервером  ");
$query = "CREATE DATABASE IF NOT EXISTS db";
$result = mysql_query ($query) or die("Ошибка при выполении запроса:
    ".mysql_error());
$db = mysql_select_db ("db",$connection) or die ("Ошибка при выборе
   базы данных  ") ;
echo " База данных'db' успешно создана ";
mysql_close($connection);
?>
</center>
</body>
</html>
```

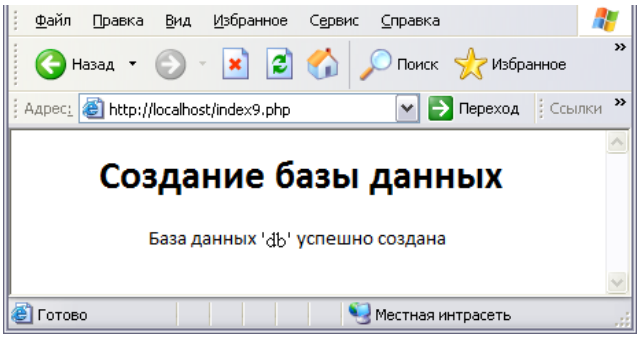


Рисунок 6.2 – Результат создания базы данных

Создание новой таблицыи

Для создания таблицы необходимо установить соединение с SQL-сервером и выбрать базу данных, в которой она будет размещена. Эти операции уже были рассмотрены выше. После этого необходимо сформировать SQL-запрос, описывающий структуру создаваемой таблицы, и выполнить его с помощью функцииmysql_query(создаем таблицу fruit, содержащую два поля – строку name и целое число number):

$query = "CREATE TABLE fruit (name VARCHAR(20), INT)";

$result = mysql_query($query) or die("Ошибка при выполнении запроса:".mysql error());

Существует большое количество типов данных, наиболее распространенные из которых:

- ✓ ВАРЧАР(length). Строка символов переменной длины, максимальная длина строки равна length.
- ✓ INT. Целое число.
- ✓ DECIMAL(totaldigits, decimalplaces). Десятное число, общее количество цифр – totaldigits, количество знаков после запятой – decimalplaces.
- ✓ DATETIME. Дата и время, например «2006-04-02 22:00:00».

Полностью скрипт для создания таблицы приведен ниже.

```
<html>
<head>
<title> Создание таблицы </title>
</head>
        <body> <center>
        <h1> Создание таблицы </h1>
        <?
        $connection = mysql_connect ("localhost", "root", "") or die (" Ошибка
соединения с сервером ");
        $db = mysql_select_db ("db", $connection) or die (" Ошибка при выборе
базы данных");
        $query = "CREATE TABLE fruit (name VARCHAR(20), number INT)" ;
        $result = mysql_query ($query) or die ("Ошибка при выполнении запроса:
".mysql_error ());
        echo " Таблица  'fruit' успешно создана ";
        mysql_close ($connection);
        ?>
        </center>
        </body>
```

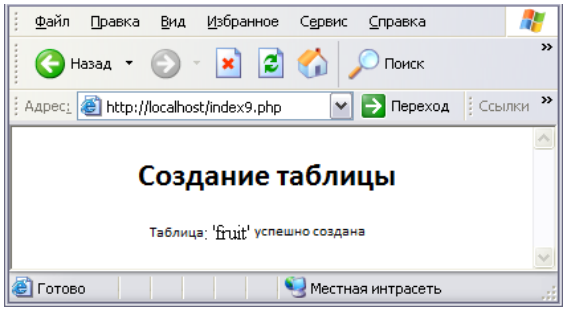


Рисунок 6.3 – Результат создания таблицы базы данных

Добавление данных

В предыдущем разделе была создана база данных db, содержащая единую таблицу fruit. Пусть необходимо занести в базу данные, приведенные в таблице. Таблица содержит два поля – name, содержащее наименование фрукта, и number, содержащее количество соответствующего фрукта.

name	*number*
яблоки	1020
груши	3329
персики	961

Чтобы наполнить ее данными, можно воспользоваться SQL-запросом INSERT. После установления соединения с SQL-сервером и выбора базы данных формируется запрос на добавление строки в таблицу, выполняемую с помощью уже знакомой функцииmysql_query:

```
$query = "INSERT INTO fruit (name, number) VALUES ('яблоки', '1020')" ;
$result = mysql_query ($query)
or die("Ошибка при выполнении запроса: ".mysql_error());
```

В следующем примере в таблицу fruit добавляются три строки, содержащие сведения о яблоках, грушах и персиках.

```
<html>
<head>
<title> Добавление данных </title>
</head>
<body> <center>
<h1> Добавление данных </h1>
<?
$connection = mysql_connect ("localhost", "root","") or die (" Ошибка
    соединения с вервером");
$db =mysql_select_db ("db",$connection) or die ("Ошибка при выборе базы
   данных ") ;
$query ="INSERT INTO fruit (name, number) VALUES ( ' яблоки', '1020')" ;
$result = mysql_query ($query) or die ("Ошибка при выполнении запроса:
    ".mysql_error ());
$query = "INSERT INTO fruit (name, number) VALUES ( ' груши ', '3329')";
$result = mysql_query ($query) or die ("Ошибка при выполнении запроса:
    ".mysql_error ());
$query = "INSERT INTO fruit (name, number) VALUES ( ' персики',
    '961')";
$result = mysql_query ($query) or die ("Ошибка при выполнении
    запроса:".mysql_error ());
echo " Данные добавлены успешно ";
mysql_close ($connection);
?>
</center> </body> </html>
```

Отображение данных

В предыдущих разделах была создана база данных и таблица fruit, содержащая три строчки. Для отображения всех строк этой таблицы в виде HTML-документа используется SQL-запрос SELECT. В результате выполнения этого запроса формируется набор данных, содержащий все строчки требуемой таблицы.

$query = "SELECT * FROM fruit";

$result = mysql_query ($query)

or die("Ошибка при выполнении запроса:" .mysql_error());

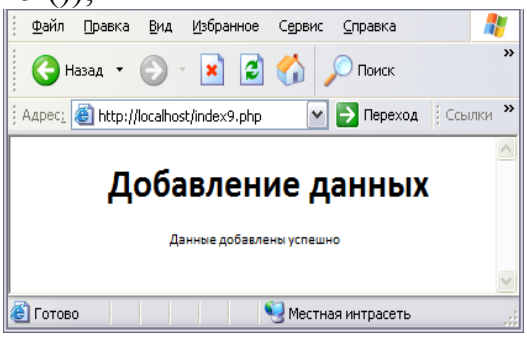


Рисунок 6.4 – Сообщение о результате добавления данных

Для последовательного отображения всех строк таблицы используется цикл while в сочетании с функциейmysql_fetch_array. Эта функция возвращает очередную строчку из набора данных в виде массива, проиндексированного именами полей таблицы:

```
while ($row = mysgl_fetch_array ($result))
{
  echo "<TD>", $row ['name'], "</TD>
   <TD>", $row ['number'] "</TD>";
  echo "<TR>";
  echo "<TD>" /TD>";
  echo "</TR>";
```

Следующий пример демонстрирует использование всех вышеперечисленных функций

```
<html>
<head>
<title> Отображение данных </title>
</head>
<body>
<center>
<h1> Отображение данных </h1>
<?
$connection = mysql_connect ("localhost", "root", "") or die (" Ошибка соеди
  нения с сервером  ");
$db = mysql_select_db ("db", $connection) or die (" Ошибка при выборе базы
      данных");
$query = "SELECT * FROM fruit";
$result = mysql_query ($query) or die ("Ошибка при выполнении запроса:
   ".mysqlerror ());
echo "<TABLE BORDER='1'>";
      echo "<TR>";
      echo "<TH> Название </TH> <TH>Количество</TH>" ;
      echo "</TR>";
      while ($row = mysql_fetch_array ($result))
      {
      echo "<TR>";
      echo "<TD>" ,$row ['name'], "</TD><TD>",  $row ['number'], "</TD>";
      echo "</TR>";
      }
      echo "</TABLE>";
      mysql_close ($connection);
      ?>

      </center>
      </body>
      </html>
```

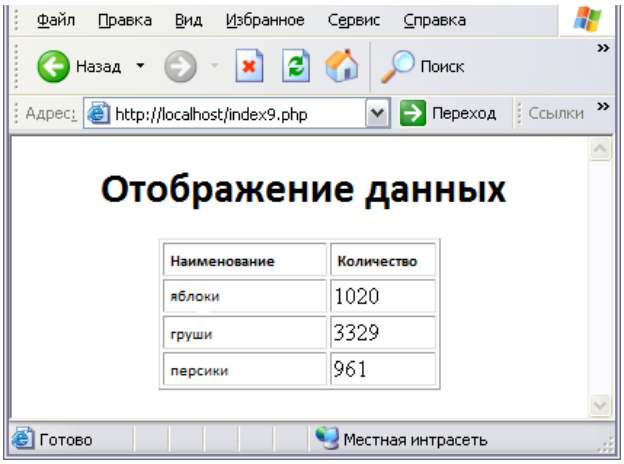


Рисунок 6.5 – Пример отображения содержимого таблицы БД

Запрос, приведенный выше, возвращает все строчки таблицы. Есть возможность выбрать только те записи таблицы, которые удовлетворяют заданному условию. Для этого используется ключевое слово WHERE. Например, для выборки всех записей, в которых в поле «name» указано «яблоки», используется следующий запрос:

SELECT * FROM fruit WHERE name="яблоки"

Кроме сравнения, в запросах можно использовать следующие операторы:

- <(меньше);
- <= (меньше или равно);
- > (больше);

• >= (больше или ровно).

Кроме того, с использованием ключевого слова IN можно задать список значений, среди которых должно находиться значение поля для выполнения условия. Например, для выборки всех записей о яблоках и апельсинах можно использовать следующий запрос:

```
SELECT * FROM fruit WHERE name IN ("яблоки", "апельсины")
```

Для построения условия можно использовать также логические операторы. Например, нужно выбрать все записи из таблицы, удовлетворяющие следующим двум условиям: в поле «name» указаны «яблоки» или «апельсины», а поле «number» содержит какое-либо значение (считается, что поле, не содержащее никакого значения) , содержит специальное значение NULL):

```
SELECT * FROM fruit WHERE name IN ("яблоки", "апельсины") AND
number IS NOT NULL
```

Для построения сложных условий используются три логических оператора – AND, OR и NOT. Если два простых условия объединены с помощью AND, то результирующее условие выполняется только в том случае, когда оба простых условия выполняются. Если же два простых условия объединены с помощью OR, то результат истинен, если хотя бы одно из простых условий истинно. Оператор NOT проводит логическую инверсию условия – истинное становится ошибочным и наоборот.

Изменение данных

Нельзя скорректировать данные, лежащие мертвым ненужным грузом. Для изменения данных нужно всего-навсего написать соответствующий SQL-запрос. К примеру, нужно изменить количество яблок в таблице fruit. Сначала следует установить соединение с базой данных аналогично тому, как это сделано в предыдущих разделах. Затем выполняется SQL-запрос, изменяющий заданную строчку в таблице:

```
$query = "UPDATE fruit SET number = 234 WHERE name = 'яблоки'";
$result = mysql_query ($query)
or die("Ошибка при выполнении запроса:" .mysql_error());
```

Наконец, измененная таблица отображается в окне обозревателя. Пример демонстрирует возможность изменения данных в базе MySQL с помощьюPHP, а результат выполнения этого кода приведен ниже.

```
<html>
<head>
<title>Изменение данных</title>
```

```
</head>
<body> <center>
<h1>Замена данных </h1>
<?
$connection = mysql_connect ("localhost", "root", "") or die ("Ошибка соеди
      нения с сервером ");
$db = mysql_select_db ("db", $connection) or die ("Ошибка при выборе ба
   зы данных");
$query = "UPDATE  fruit SET number = 234 WHERE name = ' яблоки'";
$result = mysql_query ($query) or die ("Ошибка при выполнении запроса:
   ".mysql_error ());
$query = "SELECT * FROM fruit";
$result = mysql_query ($query) or die (" Ошибка при выполнении запроса:
   ".mysql_error ());
echo "<TABLE BORDER='1'>" ;
echo "<TR>";
echo "<TH>Наименование </TH><TH>Количество </TH>" ;
echo "</TR>";
while ($row = mysql_fetch_array ($result))
{  echo "<TR>";
echo "<TD>" ,$row ['name'], "</TD><TD>",  $row ['number'], "</TD>";
echo "</TR>";  }
echo "</TABLE>";
mysql_close ($connection);
?>
</center>  </body>
</html>
```

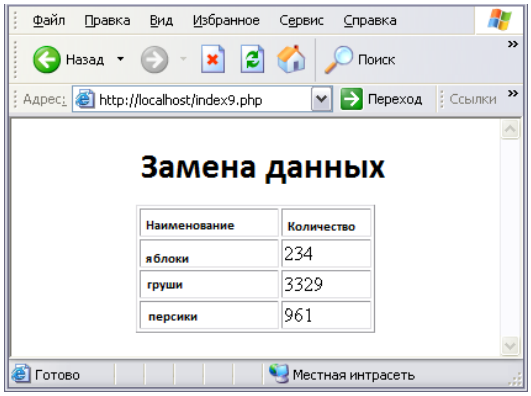


Рисунок 6.6 – Результат выполнения изменения содержимого таблицы базы данных

Сортировка данных

В вышеприведенном примере данные выводятся в том порядке, в каком они были добавлены в таблицу. Но гораздо удобнее видеть данные, упорядоченные по какому-нибудь критерию. Для этого в запросе SELECT употребляется конструкция ORDER BY. В ней задается поле, список полей или выражение, используемое для сортировки данных. Необязательный параметр DESC задает сортировку в обратном порядке. Ниже приведен фрагмент кода, возвращающий список фруктов, отсортированный по их наименованию:

```
$query = "SELECT * FROM fruit ORDER BY name";
$result = mysql_query ($query)
or die ("Ошибка при выполнении запроса: ".mysql_error());
```

Этот фрагмент использован в следующем примере.

```
<html>
<head>
<title>Сортировка данных</title>
</head>
<body>
<center>
<h1>Сортировка данных</h1>
<?
$connection = mysql_connect ("localhost", "root", "") or die ("Ошибка соеди
    нения с сервером");
$db = mysql_select_db ("db", $connection) or die ("Ошибка при выборе ба
  зы данных");
$query = "SELECT * FROM fruit ORDER BY name";
$result = mysql_query ($query) or die ("Ошибка при выполнении запроса:
    ".mysql_error ());
echo "<TABLE BORDER='1'>" ;
echo "<TR>";
echo "<TH>Наименование</TH><TH>Количество </TH>";
echo "</TR>";
while ($row = mysql_fetch_array ($result))
{
echo "<TR>";
echo "<TD>" ,$row ['name'], "</TD><TD>",  $row ['number'], "</TD>";
echo "</TR>";
}
echo "</TABLE>";
mysql_close ($connection);
?>
```

```
</center>
</body>
</html>
```

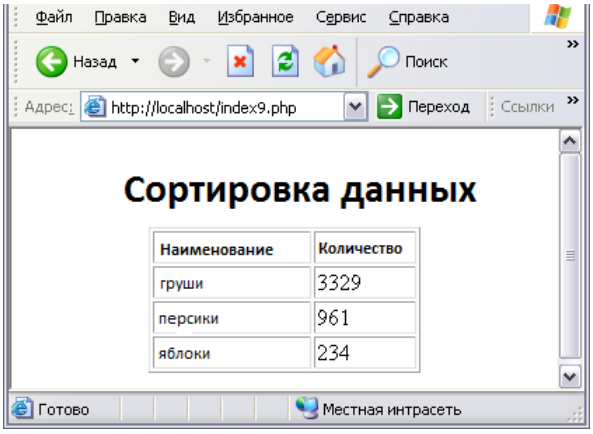


Рисунок 6.7 – Результат сортировки таблицы БД по наименованию

Удаление данных

Для удаления данных из таблицы используется SQL-запрос DELETE. В запросе обычно присутствует условие WHERE, идентифицирующее данные, которые следует удалить. Если ошибочно опустить условие, будут удалены все записи из таблицы, так что в этом случае следует быть особенно внимательным.

Например, удаление из таблицы fruit заданного фрукта естественным образом производится по его названию, как показано ниже:

```
$query = "DELETE FROM fruit WHERE name = 'яблоки'";
```

```
$result = mysql_query ($query)
or die ("Ошибка при выполнении запроса: ".mysql_error());
```

Пример демонстрирует удаление одной строки в таблице fruit и отображение всех оставшихся в ней записей. Следует обратить внимание на то, что если условии отбора строк для удаления не удовлетворила ни одна запись из таблицы, это не ошибка, а такую ситуацию следует диагностировать отдельно с помощью функции mysql_affected_rows.

```
<html>
<head>
<title>Удаление данных</title>
</head>
<body>
<center>
<h1> Сортировка данных </h1>
<?
$connection = mysql_connect ("localhost", "root", "") or die ("Ошибка соединения с
   сервером");
$db = mysql_select_db ("db", $connection) or die ("Ошибка при выборе базы дан-
   ных");
$query = "DELETE FROM fruit WHERE name = 'яблоки'";
$result = mysql_query ($query) or die ("Ошибка при выполнении запроса:
   ".mysql_error());
$query = "SELECT * FROM fruit";
$result = mysql_query($query) or die("Ошибка при выполнении запро-
   са:".mysql_error());
echo "<TABLE BORDER='1'>" ;
echo "<TR>";
echo "<TH>Наименование</TH><TH>Количество</TH>";
echo "</TR>";
while ($row = mysql_fetch_array ($result))
{echo "<TR>";
echo "<TD>" ,$row ['name'], "</TD><TD>", $row ['number'], "</TD>";
echo "</TR>"; }
echo "</TABLE>";
mysql_close($connection);
?>
</center>
</body>
```

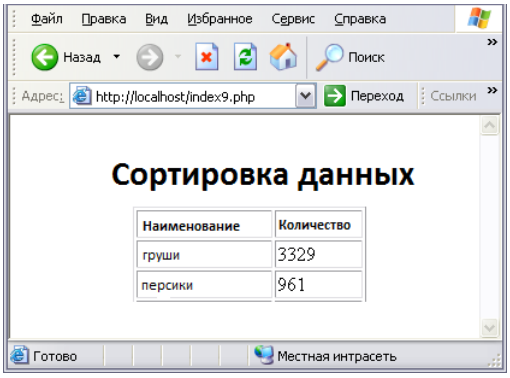


Рисунок 6.8 – Результат удаления данных из таблицы

Контрольные вопросы

1. Что такое база данных?
2. Как подключиться к MSQL?
3. Как создать базу данных?
4. Как создать таблицу в базе данных?
5. Как добавить записи в таблицу?
6. Как отобразить содержимое таблицы?
7. Как выполнить выбор элементов из таблицы?
8. Как отсортировать данные?
9. Как изменить данные в таблице?
10. Как удалить данные из таблицы?

Лабораторная работа №7

Тема:Работа с PhpMyAdmin

Цель:Научиться работать с базами данных с помощьюPhpMyAdmin

Задание для выполнения

С помощью PhpMyAdmin

1. Создайте нового пользователя с полным набором привилегий.
2. Создать в соответствии с тематикой курсовой работы базу данных, содержащую не менее 5 полей различного типа.
3. Заполнить созданную таблицу информации.
4. Создать резервную копию базы данных.
5. Восстановить базу данных из созданного дампа.

Отчет должен содержать:

1. Тему, цель работы
2. Описание проделанной работы с соответствующими скриншотами.

Методические указания

PhpMyAdmin– это набор скриптов php, предназначенных для полноценной работы с базами данных MySQL, в том числе и удаленно. Для того чтобы управлять БД и вносить в них изменения, можно обычно использовать консоль MYSQL, однако непосредственная работа с базами данных путем составления и ввода SQL-запросов требует определенной квалификации.

Возможности PhpMyAdmin

- создание и удаление баз данных
- создание, удаление, копирование, изменение и переименование таблиц
- создание, удаление и изменение полей БД
- выполнение любых валидных SQL-запросов
- управление ключами
- создание и просмотр дампов таблиц
- экспорт и импорт данные в форматах CSV, XML, PDF, Word, Excel и LATEX
- администрирование нескольких серверов
- управление пользователями MYSQL и правами доступа
- проверка целостности данных в таблицах MYISAM
- осуществление поиска в БД
- поддержка mysqli (расширение MYSQL)
- многоязычие

Для того чтобы попасть в PhpMyAdmin, запускаем ДЕНВЕР, в адресной строке браузера набираем привычное http://localhost. Прокручиваем загрузившуюся страницу в список ссылок.

URL	Описание
https://subdomain.localhost/ssl.php	Проверка SSL
http://subdomain.localhost/	Проверка "не-Интернет" доменов второго уровня, а также SSI
http://test1.ru/	Проверка "Интернет"-доменов второго уровня: test1.ru (вначале отключите прокси-сервер!)
http://subdomain.test1.ru/	Проверка "Интернет"-доменов третьего уровня
http://localhost/Tests/phpnotice/index.php	Проверка перехвата PHP Notice в Денвере
http://localhost/Tests/PHP5/index.php5	PHP5 information
http://localhost/Tools/phpMyAdmin	Проверка MySQL и phpMyAdmin
http://custom-host:8648	Проверка хоста с другим IP-адресом и портом (127.0.0.2:8648) *В Windows XP SP2 имеется ошибка, из-за которой данный хост может не работать. Официальную "заплатку" от Microsoft качайте здесь.*
http://localhost/Tests/sendmail/index.php	Проверка отладочной заглушки для sendmail

Рисунок 7.1 – Список ссылок Денвера

Переходим по ссылкеhttp://localhost/Tools/phpMyAdmin. В результате загрузится программа phpMyAdmin.

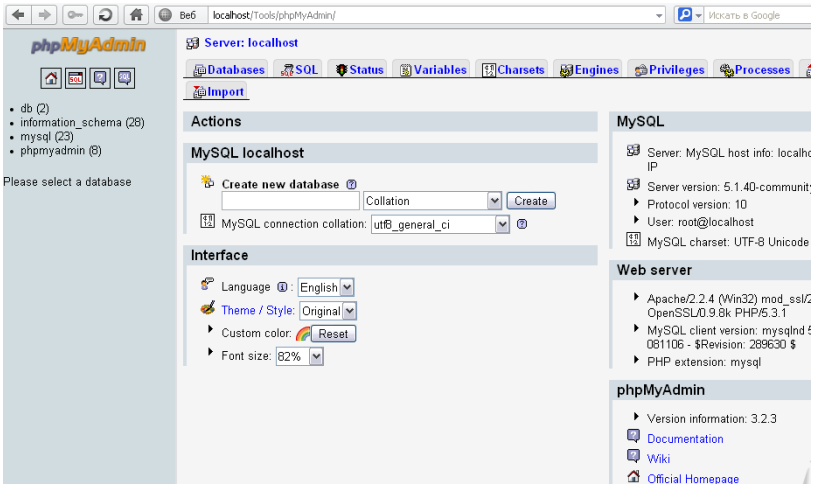


Рисунок 7.2 – Интерфейс phpMyAdmin.

В правой части страницы отображается статистическая информация об установленном программном обеспечении.

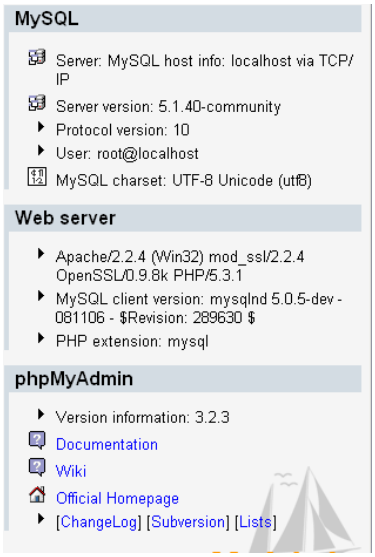


Рисунок 7.3 – Статистическая информация об установленном ПО

Чтобы перейти к списку баз данных, щелкните вкладку Databases (Базы данных).

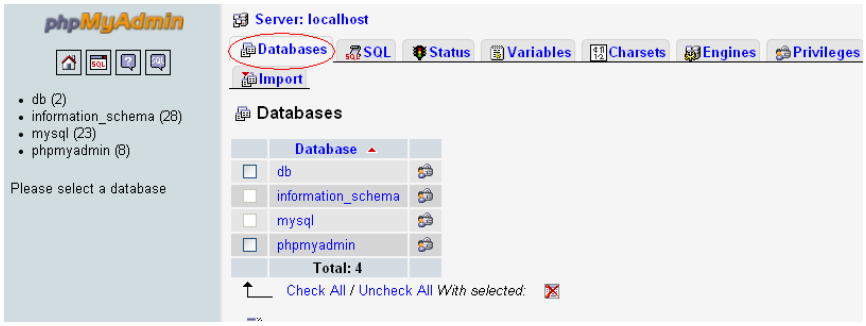


Рисунок 7.4 – Вкладка Databases (Базы данных)

Выбрать конкретную БД можно, щелкнув мышкой по ее имени. В этом случае вы увидите список всех таблиц, присутствующих в выбранной базе данных.

Концепция БД предполагает многоуровневую систему хранения данных. На верхнем уровне лежат те же базы данных. Базы данных состоят из таблиц. А уже в таблицах хранятся так называемые записи, то есть уже непосредственно конечные данные.

Выбрав галочками конкретный набор таблиц, можно делать разные операции с ними.

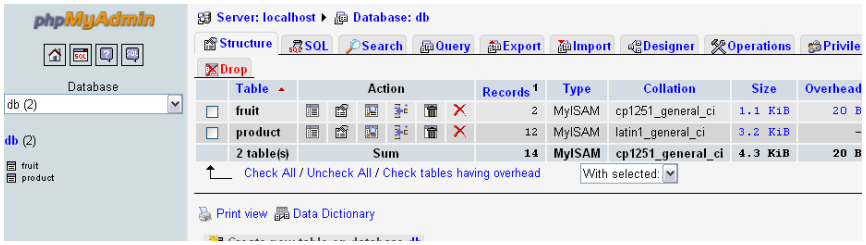


Рисунок 7.5 – Вкладка Таблицы

Кроме того, против каждой таблицы в БД находится целый набор инструментов (структура, поиск, просмотр, вставка, очистка, удаление). Внимание! Очистка таблицы и ее удаление – совершенно разные вещи. При очистке сами таблицы не удаляются, исчезает только их содержимое (записи), тогда как после удаления таблицы больше не существует.

Выбрать базу данных можно и из выпадающего списка.

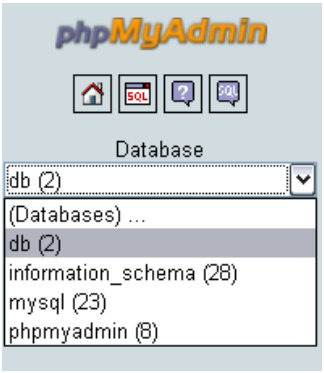


Рисунок 7.6 – Список для выбора баз данных

Создание новых пользователей и баз данных

Перед тем как создать базу данных, рассмотрим, каким образом добавляется информация о пользователях. Для этого нужно обратиться к вкладке phpMyAdmin Privileges (Привилегии).

Здесь есть список всех пользователей, имеющих те или иные права на базы данных. Чтобы создать нового пользователя, щелкните Add a new User (Создать нового пользователя).

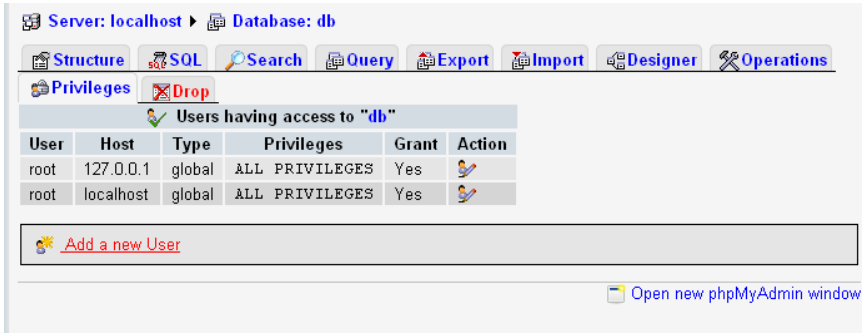


Рисунок 7.7 – Вкладка Privileges (Привилегии)

Заполняем текстовые поля с именем нового пользователя, хоста и пароля (звездочками и дважды). В случае если хост должен быть localhost, достаточно выбрать из выпадающего списка Local, как дело появится localhost. Устойчивый пароль придумывать не обязательно, но его можно сгенерировать (Generate password, Сгенерировать пароль).

Далее перейдем к созданию баз данных. Есть возможность выбрать три варианта:

None– будет создан только пользователь, новая база данных при этом не создается. Новый пользователь может быть привязан к одной из баз и получить на нее определенные права.

Create database with same name and grant all privileges– при выборе этой опции вместе с новым пользователем также будет создана и база данных, причем с тем же именем. При этом пользователь будет иметь на данную базу данных все права.

Grant all privileges on wildcard name– предоставление прав по префиксу. Данный выбор нами использоваться пока не будет.

Выбираем вариант создания БД вместе с пользователем.

Несколько ниже расположена таблица с опциями для отметки необходимых глобальных привилегий. Дело в том, что в MySQL есть несколько уровней доступа. Вообще их существует четыре:

*Глобальный уровень*применяется ко всем базам данных на указанном сервере.

*Уровень базы данных*применяется ко всем таблицам указанной базы данных.

*Уровень таблицы*применяется ко всем столбцам указанной таблицы.

*Уровень столбца*применяется к отдельным столбцам указанной таблицы.

Помните, что без особых на то оснований нет смысла предоставлять новому пользователю глобальные привилегии.

Жмем Go, получаем сообщение, что новый пользователь создан.

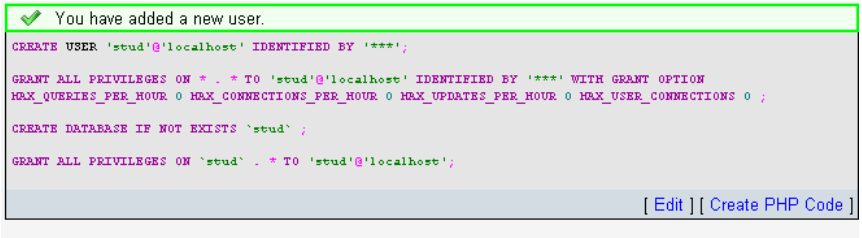


Рисунок 7. 8 – Сообщение о создании нового пользователя

Также мы видим, что появилась новая база.

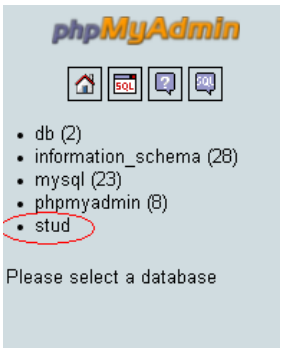


Рисунок 7.9 – Вкладка баз данных

Информация в базе данных хранится в таблицах. Поэтому нужно создать хотя бы одну таблицу с некоторым количеством полей. Для этого воспользуемся полем "Создать новую таблицу в БД".

Задаем имя таблицы и указываем количество полей в данной таблице.

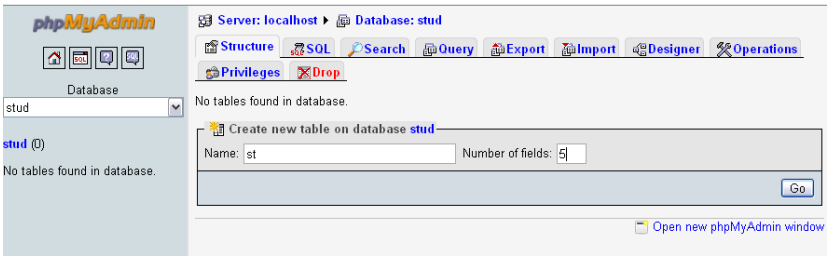


Рисунок 7.10 – Создание новой таблицы

Таблиц в базе может быть сколь угодно много. И для того чтобы взять данные из таблицы нужно будет просто указать ее название и поле, из которого будет происходить выборка данных, а дальше вставлять в нужное место HTML-каркаса с помощью php-скриптов.

После нажатия кнопки Go загружается страница создания полей в базе данных.После внесения информации сохраняем результат и создаем новую таблицу. В результате этого у нас откроется окно с настройками будущих полей.

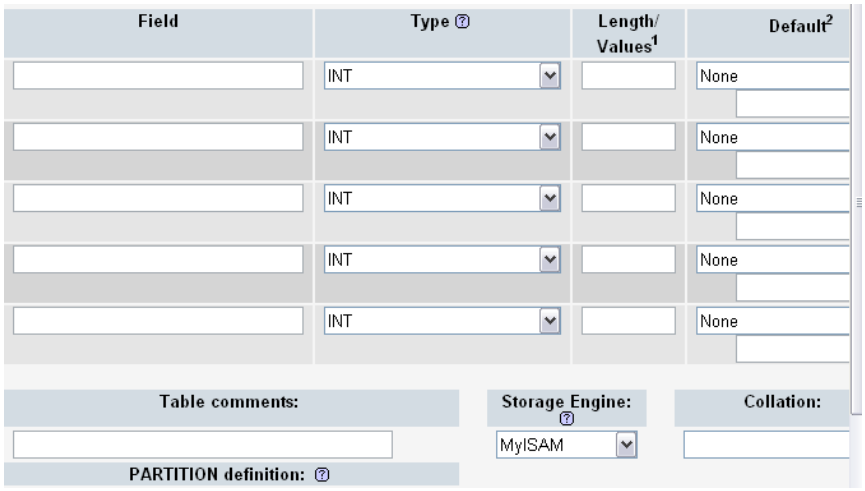


Рисунок 7.11 – Окно для задания полей базы данных

· Поле – даем имя нашим полям.

· Тип – указываем тип данных, которые будут храниться в выбранном поле.

·Длины/Значение – указываем длину будущего поля. Например, если выбран тип VARCHAR, который может хранить до 255 знаков, то в данном поле нужно будет указать длину поля от 0 до 255 символов. В нашем случае для имени и фамилии, на мой взгляд, вполне будет достаточно 100 символов.

·Сравнение – выбираем сравнение для поля. Обычно выбирается тип сравнения, который соответствует кодированию базы данных.

· Дополнительно – можно указать дополнительные свойства. Для поля можно выбрать свойство auto_increment, которое отвечает за автоматическое изменение счетчика. Также можно поставить переключатель в положение «Первоначальный ключ», говорящее о том, что в данном поле значения не могут повторяться.

После внесения всех данных сохраняем результат и создаем новые поля. Теперь при выборе таблицы слева в меню phpMyAdmin у нас откроется вкладка «Структура», в которой мы можем просмотреть всю информацию о созданных полях.

Если вы по каким-то причинам хотите добавить еще поля в таблицу, то для этого выбираем нужную нам таблицу слева в меню phpMyAdmin и в открывшемся окне ищем вкладку Add, с помощью которой можно вставить поле в начало, конец таблицы или после выбранного поля.

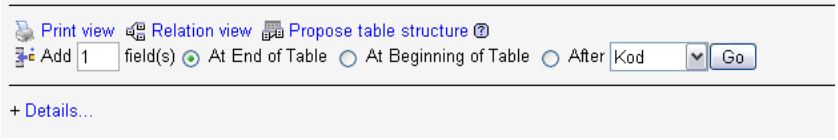


Рисунок 7.12 – Вкладка для добавления полей

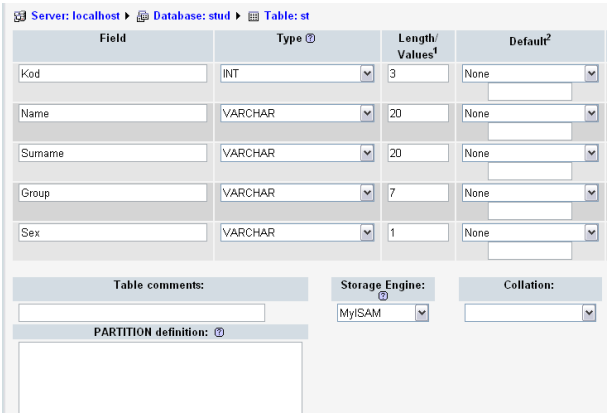


Рисунок 7.13 – Определение параметров полей базы данных

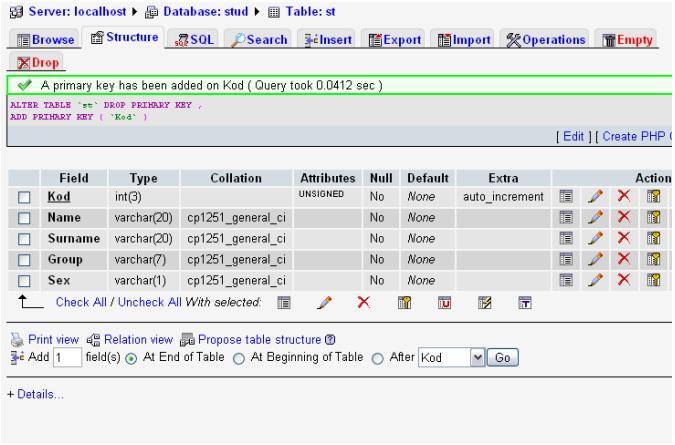


Рисунок 7.14 – Созданные поля БД

Ручное добавление информации в базу данных через phpMyAdmin

Чтобы добавить информацию в базу данных через phpMyAdmin, достаточно выбрать нужную таблицу слева в меню phpMyAdmin и перейти во вкладку «Вставить». В результате у нас откроется страница с полями для заполнения.

В поле «Значение» вносим информацию, которую мы хотим добавить в базу данных. После внесения всей необходимой информации не забывайте сохранить результат.

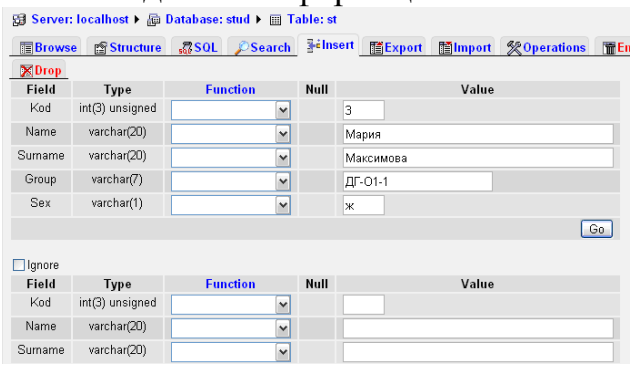


Рисунок 7.15 – Добавление данных в таблицу

После сохранения результата переходим во вкладку Обзор. В результате у нас откроется страница, на которой будут отображаться данные, находящиеся в выбранной нами таблице базы данных. В нашем случае данные будут выглядеть следующим образом.

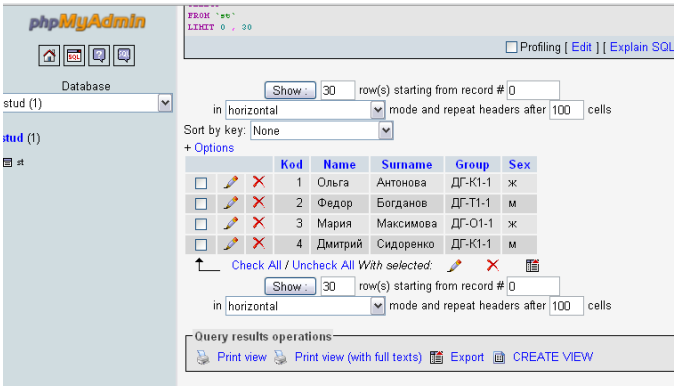


Рисунок 7.16 Заполненная таблица БД

Создание бекапа базы данных

PhpMyAdmin позволяет создавать копию базы данных, а также восстанавливать БД из бекапа. Процесс создания копии достаточно прост. Для начала нужно выбрать базу данных, резервную копию которой необходимо создать. Далее пользуемся вкладкой Export (Экспорт), перед нами открывается окно, в котором нужно установить необходимые параметры:

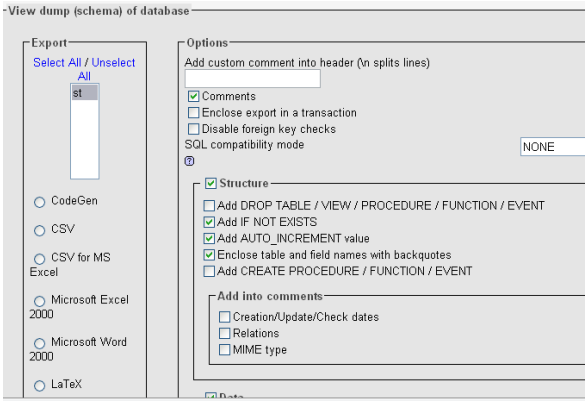


Рисунок 7.17 – Вкладка Export (Экспорт),

А вот в последней части окна нужно отметить опцию Save as file (сохранить в файл). Если этого не сделать, то вся информация с базой будет отправлена на консоль phpMyAdmin.

Если же флажок поставить и указать имя файла, то дамп базы будет сохранен в файл с этим названием.

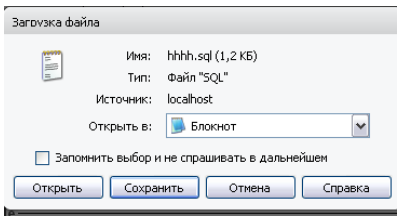


Рисунок 7.18 – Окно загрузки файла

При сохранении можно указать использование компрессии и ее тип. Если дамп базы данных невелик, то разницы практически не будет. Однако, если имеем дело с объ-

емной базой данных, то имеет смысл сжать дамп, тем более что текстовые данные сжимаются хорошо.

Восстановление базы данных по дампе

Для восстановления базы данных используется созданный дамп, прилагаемый на вкладке Import (Импорт). Он применяется к созданной пустой базе данных.

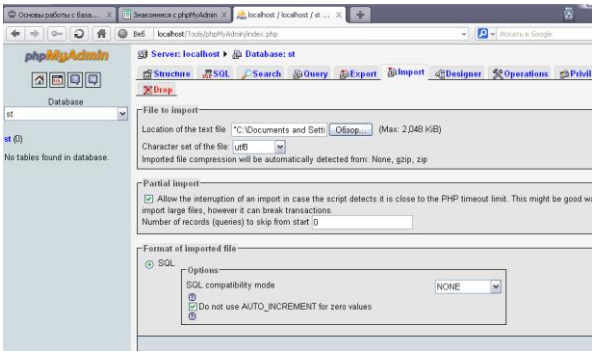


Рисунок 7.19 – Вкладка Import (Импорт)

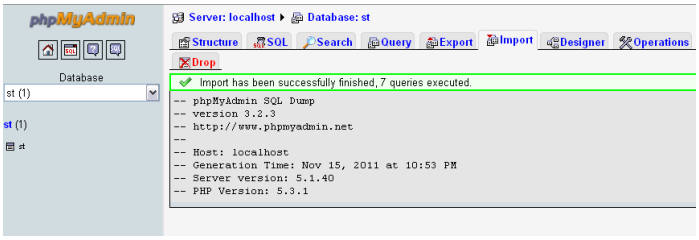


Рисунок 7.20 – Сообщение об успешном восстановлении

Контрольные вопросы

1. Что такоеPhpMyAdmin? Его способности.
2. Как скачать PhpMyAdmin?
3. Как создать нового пользователя?
4. Какие привилегии может иметь пользователь?
5. Как создать новую базу данных?
6. Какие параметры полей базы данных можно указать?
7. Как добавить данные в таблицу базы данных?
8. Как удалить данные из таблицы и саму таблицу?
9. Как создать дамп базы данных?
10. Как восстановить базу данных по дампе?

Лабораторная работа №8

Тема:PHP. Работа с базами данных

Цель:Научиться создавать сценарии на языкеPHP
с использованием MySQL

Задание для выполнения

Для сайта курсовой работы

а) Добавить форму для входа и регистрации. Данные должны храниться в базе данных.

б) создать базу данных продуктов и форму для заказа продуктов из данной базы. Предусмотреть подсчет общей суммы заказа.

Отчет должен содержать:

1. Тему, цель работы
2. Коды созданных файлов и копию окна обозревателя при их просмотре.

Методические указания

Рассмотрим простые действия с базой данных – добавление записей и отображение содержимого таблиц. В качестве примера будем использовать базу данных TVR с таблицей tovar, содержащую информацию о наличии товаров на складе - их код, наименование, единицу измерения, цену за единицу, количество каждого наименования. Как правило, с аналогичными таблицами имеют дело менеджеры интернет-магазинов.

Структура базы данных

Название	*Название поля*	*Тип*
Код	kod	integer
Наименование	name	varchar
Единица измерения	ed	varchar
Количество	quantity	integer
Цена за единицу	price	float

Построение интерфейса для добавления информации

Первое, что нужно сделать – это установить соединение с базой данных. Для этого используется функцияmysql_connect.

Синтаксис ресурс mysql_connect ([строка server[, строка username
[, строка password [, логическое new_link [, целое client_flags]]]]])

Данная функция устанавливает соединение с сервером MSQL и возвращает указатель на это соединение или FALSE в случае неудачи. Если функция вызывается дважды с одними и теми же параметрами, новое соединение не устанавливается, а возвращается ссылка на старое соединение. Чтобы этого избежать, используют параметр new_link, заставляющий в любом случае открыть еще одно соединение. Соединение с сервером закрывается при завершении выполнения скрипта, если оно до этого не было закрыто с помощью функцииmysql_close().

В нашем случае устанавливаем соединение с базой данных на локальном пользовательском сервере menedger с паролем "new".

После этого соединения нужно выбрать базу данных, с которой будем работать. В PHP выбор базы данных осуществляется с помощью логической функцииmysql_select_db.

Синтаксис mysql_select_db (строка database_name[, ресурс link_identifier])

Эта функция возвращает TRUE при успешном выборе базы данных и FALSE – в противном случае.

Далее следует получить перечень полей таблицы, их наименования, длины, типы и флаги и вывести их в соответствующие поля html-формы.

Функция
mysql_list_fields (строка database_name, строка table_name
[, ресурс link_identifier])

возвращает список полей в таблице table_name в базе данных database_name. Результат работы этой функции – переменная типа ресурса. Это ссылка, которую можно использовать для получения информации о полях таблицы, включая их названия, типы и флаги.

Функцияmysql_field_nameвозвращает имя поля, полученного в результате исполнения запроса. Функцияmysql_field_lenвозвращает длину поля. Функцияmysql_field_typeвозвращает тип поля, а функцияmysql_field_flagsвозвращает список флагов поля, записанных через пробел. Типы поля могут быть int, real, string, blob и т.д. Флаги могут быть not_null, primary_key, unique_key, blob, auto_increment и т.д.

Синтаксис всех этих команд одинаков:
строка mysql_field_name (ресурс result, целое field_offset)
строка mysql_field_type (ресурс result, целое field_offset)
строка mysql_field_flags (ресурс result, целое field_offset)
строка mysql_field_len (ресурс result, целое field_offset)

Здесь result – это идентификатор результата mysql_list_fields или mysql_query, а field_offset – порядковый номер поля в результате.

Используя эти функции, получаем перечень полей таблицы и отображаем их в html-форму. Начальный код скрипта приведен в листинге 1, а результат отображения в браузере представлен на рис. 10.1

Листинг 1.

```
<? $conn=mysql_connect("localhost", "menedger", "new");
// устанавливаем соединение
```

```
$database = "TVR"; $table_name = "tovar";
mysql_select_db($database); // выбираем базу данных для работы
$list_f = mysql_list_fields($database,$table_name);
// получаем список полей в базе
$n = mysql_num_fields($list_f); // число строк в результате
// предыдущего запроса (т.е. сколько всего полей в таблице tovar)
echo "<form method=post action=add.php>";
// создаем форму для ввода данных
echo "  <table border=0 cellspacing=0 width=40% >
<tr> <td bgcolor='#005533' align=center><font color='#DDDDDD'>
<b> Добавить новую запись в таблицу $table_name</b></font></td></tr> </table>";
echo "<table border=0 cellspacing=1 cellpadding=0 width=50% >";
// для каждого поля получаем его имя, тип, длину и флаги
for($i=0;$i<$n; $i++)
{ $type = mysql_field_type($list_f, $i);
$name_f = mysql_field_name ($list_f,$i);
$len = mysql_field_len($list_f, $i);
$flags_str = mysql_field_flags ($list_f, $i);
// из строки флагов делаем массив,
// где каждый элемент массива – флаг поля
$flags = explode(" ", $flags_str);
foreach ($flags as $f){
if ($f == 'auto_increment') $key = $name_f; }
/* для каждого поля, не являющегося автоинкрементом, в
В зависимости от его типа выводим подходящий элемент формы */
if ($key <> $name_f)
{ echo "<tr><td align=right bgcolor='#C2E3B6'><font size=2>
<center> <b> ". $name_f ."</b> </center> </font></td>";
switch ($type)
{ case "string": $w = $len;
echo "<td><input type=text name=\"$name_f\"
size = $w ></td>"; break;
case "int": $w = $len; echo "<td><input type=text name=\"$name_f\"
size = $w ></td>"; break;
case "real": $w = 10;
echo "<td> <input type=text name=\"$name_f\"
size = $w > </td>"; break;
case "blob": echo "<td><textarea rows=6 cols=60
name=\"$name_f\"></textarea></td>";
break; } }
echo "</tr>"; }
echo "</table>";
echo "       <input type=submit name='add' value='Добавить'>";
echo "       <input type=reset name='no' value='Очистить'>";
echo "</form>";
?>
```

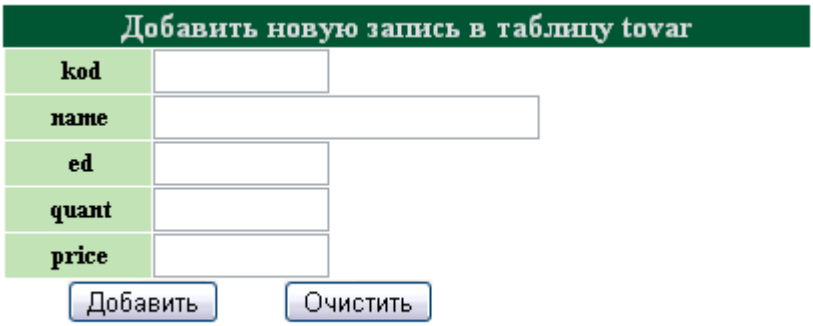


Рисунок 8.1 – Форма для добавления данных в таблицу.

Форма создана. Теперь нужно сделать самое главное – отправить данные из этой формы в таблицу базы данных. Чтобы записать данные в таблицу, используется команда INSERT языка SQL. Чтобы ею можно было воспользоваться (как и любой другой командой SQL) в PHP используется функцияmysql_query().

Синтаксис ресурс mysql_query (строка query [, ресурс link_identifier])

mysql_query() посылает SQL-запрос активной базе данных MSQL сервера, который определяется с помощью указателя link_identifier (это ссылка на какое-то соединение с сервером MSQL). Если параметр link_identifier опущен, используется последнее открытое соединение. Если отсутствуют открытые соединения, функция пытается соединиться с СУБД, аналогично функцииmysql_connect()без параметров. Результат запроса буферизуется.

Для запросов SELECT, SHOW, EXPLAIN, DESCRIBE mysql_query() возвращает указатель на результат запроса или FALSE, если запрос не был выполнен. В остальных случаях mysql_query() возвращает TRUE, если запрос выполнен успешно, и FALSE – в случае ошибки.

Скрипт add.php, приведенный в листинге 2, добавляет содержимое формы в таблицу базы данных. При этом данные из формы будут передаваться как переменные виду $_POST['имя_поля'].

Листинг 2.

```
<? $conn=mysql_connect("localhost", "menedger", "new");
// устанавливаем соединение
$database = "TVR"; $table_name = "tovar";
mysql_select_db($database); // выбираем базу данных
$list_f = mysql_list_fields($database,$table_name);
// получаем список полей в базе
$n = mysql_num_fields($list_f);
// число строк в результате предварительного запроса
// составим один запрос сразу для всех полей таблицы
$sql = "insert в $table_name set"; // начинаем создавать запрос, перебираем все поля
for($i=0;$i<$n; $i++)
{$name_f = mysql_field_name ($list_f,$i); // имя поля
$value = $_POST[$name_f]; // значение поля
$j = $i + 1;
```

```
$sql = $sql . $name_f." = '$value'";
// дописываем в строку $sql пару имя=значение

if ($j <> $n) $sql = $sql . ",";
// если поле не последнее в списке, то ставим комку
}
$result = mysql_query($sql,$conn); // отправляем запрос
// выводим сообщение успешно ли выполнен запрос
if (!$result) echo "Can't add ($table_name)";
else echo "Success!<br>";
?>
```

Чтобы отобразить данные из таблиц базы в браузере с помощью PHP, нужно сначала получить эти данные переменных PHP. При работе с MSQL выборка данных проводится с помощью команды SELECT языка SQL.

Как рассматривалось ранее, любой запрос, в том числе и по выборке, можно отправить на сервер с помощью функцииmysql_query().Результатом работы этой функции является одно из выражений, TRUE или FALSE. В данном случае нужно отправить запрос на выбор всех полей, а результат отразить в браузере. И результатом должна быть целая таблица значений, а точнее указатель на эту таблицу. В таких ситуациях используются аналоги функции.mysql_field_name()для выбора значения поля. Таких функций у PHP несколько. Наиболее популярные –mysql_result()иmysql_fetch_array().

Синтаксис смешанное mysql_result (ресурс result, целое row[, смешанное field])

mysql_result()возвращает значение одной ячейки результата запроса. Аргумент field может быть порядковым номером поля в результате, именем поля или именем поля с именем таблицы через точку tablename.fieldname. Если для имени поля в запросе применялся алиас ('select foo as bar from...'), он используется вместо реального имени поля.

Работая с большими массивами данных, следует задействовать одну из функций, обрабатывающих сразу целый ряд результатов – строку или массив (например, mysql_fetch_row(), mysql_fetch_array() и т.д.). Поскольку эти функции возвращают значение нескольких ячеек сразу, они работают гораздо быстрее mysql_result().

Синтаксис массив mysql_fetch_array (ресурс result[, целое result_type])

Эта функция обрабатывает ряд результата запроса, возвращая массив (ассоциативный, численный или оба) с обработанным рядом результата запроса или FALSE, если рядов больше нет.

mysql_fetch_array() – это расширенная версия функции mysql_fetch_row(). Помимо хранения значений в массиве с многочисленными индексами, функция возвращает значение в массиве с индексами по названию колонок.

Код CPP-скрипта для отображения данных из таблицы в окно браузера приведен в листинге 3, а результат его работы – на рис. 8.2. Рассмотрены два варианта запроса на выборку – с помощью функций mysql_result и mysql_fetch_array

Листинг 3.

```
<?
$conn=mysql_connect("localhost", "menedger", "new");
$database = "TVR"; $table_name = "tovar";
mysql_select_db($database);
$list_f = mysql_list_fields($database,$table_name);
$n1 = mysql_num_fields($list_f);
// сохраним имена полей в массиве $names
for($j=0;$j<$n1; $j++){ $names[ ] = mysql_field_name ($list_f,$j); }
$sql = "select * from $table_name"; // создаем SQL запрос
$q = mysql_query($sql,$conn) или die(); // отправляем запрос на сервер
$n = mysql_num_rows($q); // получаем число срока результата
// рисуем HTML-таблицу
echo " <table border=0 cellspacing=0 width=50%
align=center><tr><td bgcolor='#005533' align=center>
<font color='#FFFFFF'><b>$table_name</b></font></td> </tr></table>";
echo "<table cellspacing=0 cellpadding=1 border=1 width=50% align=center>";
// отображаем названия полей
echo "<tr>";
foreach ($names as $val)
{ echo "<th align=center bgcolor='#C2E3B6'>
<font size=2>$val</font></th>"; }

// Вариант 1
// отображаем значения полей
echo "</tr>";
for($i=0;$i<$n; $i++)
{ // перебираем все сроки в результате запроса на выборку
echo "<tr>";
foreach ($names as $val) { // перебираем все имена полей
$value = mysql_result($q,$i,$val); // получаем значение поля
echo "<td><font size=2> $value</font></td>";
// выводим значение поля
} echo "</tr>";
} echo "</table>";
?>
// Вариант 2
// отображаем значения полей
// получаем значение поля в виде ассоциативного массива
while($row = mysql_fetch_array($q, MYSQL_ASSOC)) {
echo "<tr>";
foreach ($names as $val){
echo "<td><font size=2> $row[$val]</font></td>";
// выводим значение поля
} echo "</tr>";
} echo "</table>";
?>
```

tovar				
kod	name	ed	quant	price
123	Конфеты "Птичье молоко"	пачка 200	100	12.5
333	Печенье "33 коровы молоко "	пачка 200	50	2.3
454	Конфеты "Батончик детский "	кг	24	14.2
676	Пирожное трубочки	кг	25	20.45
777	Вафли "Артек"	кг	12	18.6

Рисунок 8.2 – Отображение содержимого таблицы БД в браузере

Таким образом, описанная технология взаимодействия PHP и MySQL может использоваться при решении целого ряда схожих задач, таких как задачи изменения и удаления данных, задачи манипулирования таблицами базы данных и т.д. Все это типичные задачи, возникающие при разработке систем управления данными, и умение их решать, как и умение работать с базами данных в целом, очень важно для web-программиста.

Контрольные вопросы

1. Что такое база данных?
2. Как подключиться к MySQL?
3. Как создать базу данных?
4. Как создать таблицу в базе данных?
5. Как добавить записи в таблицу?
6. Как отобразить содержимое таблицы?
7. Как выполнить выбор элементов из таблицы?
8. Как отсортировать данные?
9. Как изменить данные в таблице?
10. Как удалить данные из таблицы?

Лабораторная работа №9

Тема:PHP. Работа с сессиями

Цель:Научиться создавать сценарии на языкеPHP, работающие с сеансами (сессиями)

Задание для выполнения

1. Реализовать исполнение скриптов из листингов 1, 2, 3.

2. Для главной страницы сайта курсовой работы организовать статистику посещений.

Отчет должен содержать:

1. Тему, цель работы
2. Коды созданных файлов и копию окна обозревателя при их просмотре.

Методические указания

Управление сеансами в PHP

Что такое управление сеансом |HTTP иногда называют "протоколом без состояния". Это означает, что данный протокол не имеет встроенного способа поддержания состояния между двумя транзакциями. Когда пользователь приглашает друг за другом две страницы, HTTP не обеспечивает возможности сообщить, что оба запроса будут исходить от одного и того же пользователя. Таким образом, идея управления сеансами состоит в обеспечении отслеживания пользователя в течение одного сеанса связи с Web-сайтом. Если это удастся осуществить, мы сможем легко поддерживать пользовательское подключение и предоставление ему содержимого сайта в соответствии с его уровнем прав доступа или персональными параметрами. Мы сможем отслеживать поведение пользователя.

В первых версиях CPH управление сеансами осуществлялось средствами PHPLib, базовой библиотеки PHP, которая и сейчас является полезным набором инструментов.

Четвертая и последующая версия PHP включает собственные встроенные функции управления сеансом.

Основные функциональные средства управления сеансом.Для запуска сеанса в PHP используется уникальный идентификатор сеанса SID, являющийся зашифрованным случайным числом. Идентификатор сеанса генерируется PHP и сохраняется на стороне клиента в течение всего времени сеанса. Для хранения идентификатора сеанса используется либо cookie- набор на компьютере пользователя, либо URL.

Идентификатор сеанса играет роль ключа, обеспечивающего возможность регистрации некоторых специфических переменных, так называемых переменных сеанса. Содержимое этих переменных хранится на сервере. Единственной информацией, видимой на стороне клиента, является идентификатор сеанса. Если при определенном

подключении к вашему сайту идентификатор сеанса является "видимым" либо в cookie-наборе, либо в URL, есть возможность получить доступ к переменным сеансам, которые сохранены на сервере для данного сеанса. По умолчанию переменные сеанса хранятся в двумерных файлах на сервере (при желании способ хранения можно изменить и использовать вместо двумерного файла базу данных, но для этого нужно будет написать собственную функцию).

Скорее всего, придется столкнуться с Web-сайтами, на которых для хранения идентификатора сеанса используется URL. Если в URL есть строка данных, которые выглядят случайными, то это, скорее всего, свидетельствует об использовании одной из двух описанных здесь разновидностей управления сеансом.

Другим решением проблемы сохранения состояния в течение некоторого количества транзакций, при наличии чистого внешнего вида URL, являются cookie-наборы.

Что такое cookie набор?cookie-набор – это небольшой фрагмент информации, который хранятся на клиентской машине. Чтобы установить cookie-набор на пользовательской машине, необходимо отправить ему HTTP-заголовок, содержащий данные в следующем формате.

Set-Cookie: NAME=VALUE; [expires=DATE;] [path=PATH;]
[domain=DOMAIN_NAME;] [secure]

Это создаст cookie набор с именем NAME и значением VALUE. Остальные параметры являются необязательными. В expires задается дата истечения срока действия, после наступления которой cookie-набор перестанет рассматриваться как актуальный. Отметим, что если дата истечения срока действия не задана, cookie-набор будет постоянным, пока его кто-нибудь не удалит вручную – либо администратор, либо сам пользователь). Два параметра path и domain применяются для определения одного или нескольких URL, к которым относится данный cookie-набор. Ключевое слово secure означает, что cookie набор не может отправляться через простое HTTP-соединение.

Когда браузер соединяется с URL, он сначала ищет куки-наборы, хранящиеся локально. Если какие-либо из них относятся к URL, с которым установлено соединение, они передаются обратно на сервер.

Установка cookie-наборов из PHP.cookie-наборы в PHP можно установить вручную, используя функцию setcookie(). Она имеет следующий прототип:

int setcookie (string name [, string value [, int expire [, string path [, string domain [, int secure]]]]]))

Параметры точно соответствуют тем, которые используются в описанном выше заголовке Set-Cookie. Если cookie-набор установлен как

setcookie ("mycookie", "value");

то когда пользователь обращается к следующей странице на вашем сайте (или перезагружает текущую страницу), вы получаете доступ к переменной с именем $mycookie, содержащей значение "value". Доступ к этой переменной также можно получить через $HTTP_COOKIE_VARS["mycookie"].

Для удаления cookie-набора нужно вызвать setcookie() с тем же именованием, но без указания значения. Если cookie набор устанавливался с другими параметрами (такими как специфические URL или даты окончания), нужно будет отправить те же параметры повторно, иначе cookie набор удален не будет.

Для установки cookie-набора вручную можно также воспользоваться функцией Header() и описанным выше синтаксисом представления cookie-набора. Однако при этом следует иметь в виду, что заголовки cookie наборов должны отправляться перед всеми другими заголовками (иначе заголовки cookie наборов работать не будут).

Использование cookie наборов в сеансах.При использовании cookie-наборов возникают некоторые проблемы: есть браузеры, которые не принимают cookie-наборы, а есть пользователи, запрещающие использование cookie-наборов в своих браузерах. Это одна из причин, по которым в сеансах PHP используются двойной метод cookie-набор/адрес URL.

В сеансе PHP нет необходимости задавать cookie-наборы вручную. Это за вас сделают функции сеанса. Для просмотра содержимого cookie-набора, установленного при управлении сеансом, можно воспользоваться функциейsession_get_cookie_params(). Она возвращает ассоциативный массив, содержащий элементы lifetime, path и domain.

Можно использовать также:

session_set_cookie_params($lifetime, $path, $domain);

Этот оператор устанавливает параметры cookie-набора для сеанса.

Сохранение идентификатора сеанса.В СРН cookie-наборы в сеансах используются по умолчанию. Если можно установить cookie-наборы, то для сохранения идентификатора сеанса будет использоваться именно этот способ. Другой способ, который может использоваться в PHP, заключается в добавлении идентификатора сеанса к адресу URL. Можно сделать так, чтобы идентификатор сеанса прилагался к URL автоматически – для этого следует скомпилировать PHP с опцией --enable-trans-sid.

Можно встроить идентификатор сеанса в ссылку для обеспечения его передачи. Идентификатор сеанса будет запоминаться в константе SID. Для того чтобы передать его вручную, его нужно будет добавить в конец ссылки, аналогично параметру GET:

<A HREF="link.php?<?=SID?>">

В общем случае проще компилировать PHP --enable-trans-sid, если только это возможно (отметим попутно, что константа SID может использоваться для вышеописанных целей только в том случае, если конфигурация PHP выполнялась из --эпабл-track-vars).

Реализация управления обычным сеансом.Основными этапами использования сеанса являются следующие:

– Запуск сеанса.

– Регистрация переменных сеанса.

– Использование переменных сеанса.

– отмена регистрации переменных и закрытие сеанса.

Отметим, что все перечисленные этапы не обязательно могут содержаться в одном сценарии и некоторые из них могут находиться в нескольких сценариях. Рассмотрим каждый из этих этапов последовательно.

Запуск сеанса.Прежде чем можно будет использовать функциональные возможности сеанса, следует запустить сам сеанс. Есть три способа сделать это.

Первый (и самый простой) состоит в том, что сценарий начинается с вызова функции

session_start() ;

Эта функция проверяет, существует ли идентификатор текущего сеанса. Если нет, то она его создает. Если идентификатор текущего сеанса уже существует, он загружает зарегистрированные переменные сеанса, чтобы они стали доступными для использования.

Второй способ состоит в том, что сеанс запускается при попытке зарегистрировать переменные сеанса.

Третий способ запустить сеанс – задать настройки PHP, при которых сеанс будет запускаться автоматически, как только кто-нибудь посетит ваш сайт. Для этого следует воспользоваться опциейsession.auto_startв файле php.ini.

Регистрация переменных сеанса.Для того чтобы получить возможность отслеживать переменные от одного сценария другому, их необходимо зарегистрировать. Это делается путем вызова функцииsession_register(). К примеру, для регистрации переменной $myvar применяется следующий код:

$myvar = 5; session_register("myvar");

Обратите внимание: вы должны быть переданы в функциюsession_register() строка, содержащая имя переменной. Эта строка не должна включать символ $. Данный оператор регистрирует имя переменной и отслеживает ее значение. Отслеживание переменной будет производиться, пока не завершится сеанс или пока вручную не отменится его регистрация.

За один прием можно зарегистрировать более одной переменной, передав разделенный запятыми список имен переменных:

session_register("myvar1", "myvar2");

Использование переменных сеансов.Чтобы сделать переменную сеанса доступной для использования, сначала необходимо запустить сеанс, воспользовавшись одним из описанных выше способов. После этого появляется доступ к этой переменной. Если опцияregister_globalsвключена, то доступ к этой переменной можно получить через сокращенную форму ее имени, например $myvar. Если же упомянутая опция не включена, получить доступ к переменной можно через ассоциативный массив $HTTP_SESSION_VARS, например

$HTTP_SESSION_VARS ["myvar"].

Переменные сеанса не могут быть перезаписаны данными GET или POST. Это хорошо с точки зрения обеспечения безопасности, но связано с некоторыми ограничениями при кодировании.

С другой стороны, потребуется тщательность при проверке на предмет того, установлены ли уже переменные сеанса (например, с использованиемisset() илиempty()). Кроме того, следует иметь в виду, что переменные могут быть установлены пользователем через GET или POST. Проверьте, есть ли переменная зарегистрированной переменной сеанса, можно обратившись к функцииsession_is_registered(). Вызов функции выполняется следующим образом:

$result = session_is_registered("myvar");

Эта функция проверит, является ли $myvar зарегистрированной переменной сеанса и вернет true или false. Можно поступить и по-другому – проверить массив $HTTP_SESSION_VARS на предмет наличия в нем переменной.

Отмена регистрации переменных и завершение сеанса.После окончания работы с переменной сеанса ее регистрацию можно отменить, воспользовавшись функцией

session_unregister("myvar");

Подобно функции регистрации, эта функция требует указания имени переменной, регистрацию которой необходимо отменить, в виде строки, не включающей символ $. Данная функция одномоментно может отменить регистрацию только одного переменного сеанса (в противоположностьsession_register()). Однако для отмены регистрации всех переменных текущего сеанса можно обратиться кsession_unset().

После окончания сеанса сначала нужно будет отменить регистрацию всех переменных, а затем для обнуления идентификатора сеанса вызвать

session_destroy();

Пример обычного сеанса.Вышеизложенный материал может показаться несколько абстрактным, поэтому рассмотрим пример. Приведенный в нем код обеспечивает обработку трех страниц. На первой странице запускаем сеанс и зарегистрируем переменную $sess_var. Код показан в листинге 1 ниже.

*Листинг*1. page1.php — запуск сеанса и регистрация переменной

```
<?
session_start();
session_register("sess_var");
$sess_var = "Hello world!";
echo "Наименование \ $sess_var is $sess_var<br>" ;
?>
```

Мы зарегистрировали переменную и установили ее значение.

Отметим, что мы изменили значение переменной уже после регистрации. Можно, однако, сделать и наоборот – установить значение, а затем зарегистрировать переменную. Конечное значение переменной на странице – это значение, которое будет доступно на следующих страницах. В конце сценария переменная сеанса превратится в последовательную форму (сериализируется), или замораживается, до своей перезагрузки через следующий вызовsession_start(). Таким образом, следующий сценарий начинается с вызоваsession_start(). Сценарий показан в листинге 2.

Листинг 2. page2.php – получение доступа к переменному сеансу и отмена регистрации

```
<?
session_start();
echo "Наименование \ $sess_var is $sess_var<br>";
session_unregister("sess_var");
?>
```

После вызоваsession_start() переменная $sess_var станет доступной, а ее значением будет то, которое сохранено в предыдущем сеансе.

Совершив с переменной все необходимые действия, мы вызываемsession_unregister() для отмены ее регистрации. Обратите внимание: сеанс еще существует, но переменная $sess_var уже больше не зарегистрирована.

И, наконец, мы переходим к page3.php, последнему сценарию в данном примере.

Листинг 3. page3.php- завершение сеанса

```
<?
session_start();
echo "Наименование \ $sess_var is $sess_var<br>";
session_destroy() ;
?>
```

Как видно, доступа к значению $sess_var больше нет. И в завершение – вызовsession_destroy() для разрушения идентификатора сеанса.

Конфигурация управления сеансом.Втабл. 9.1 перечисляются некоторые из наиболее полезных опций, которые можно установить в своем файле php.ini вместе с их кратким описанием.

Таблица 9.1 –Функции конфигурации сеанса

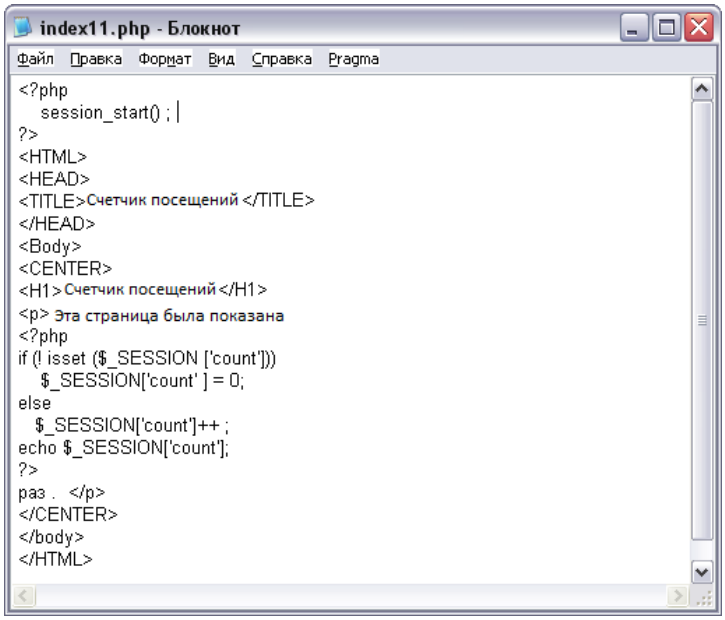
Имя опции	*Значение по умолчанию*	*Действие*
session.auto_start	0 (запретить)	Автоматический запуск сеансов.
session.cache_expire	180	Установка времени жизни для кэшированных страниц сеанса (в минутах).
session.cookie_domain	None	Домен для установки в куки-наборе сеанса.
session.cookie_lifetime	0	Определение длительности существования cookie-набора идентификатора сеанса на пользовательской машине. По умолчанию 0 – пока не будет закрыт браузер.
session.cookie_pafh	/	Путь для установки в куки-наборе сеанса.
session.name	PHPSESSID	Имя сеанса, используемое в системе пользователя как имя cookie-набора.
session.save_handler	Файлы	Определение места хранения данных сеанса. Здесь можно указать базу данных,

		однако для этого нужно будет реализовать свои функции.
session.save_path	/tmp	Путь к месту хранения данных сеанса. В более общем случае для определения и обработки передаваемых на хранение аргументов используется session.save_handler.
session.use_cookies	1 (удовлетворить)	Конфигурация сеанса с возможностью использования cookie наборов на стороне клиента.

Сохранение данных сеансов

Для хранения данных сеансов используется массив $_SESSION. Он является суперглобальным подобно $_GET, $_POST и $_REQUEST. Перед использованием массива $_SESSION следует провести инициализацию сеанса с помощью функции session_ start.

Пример использования массива $_SESSION для сохранения данных о посещении сайта приведен ниже.



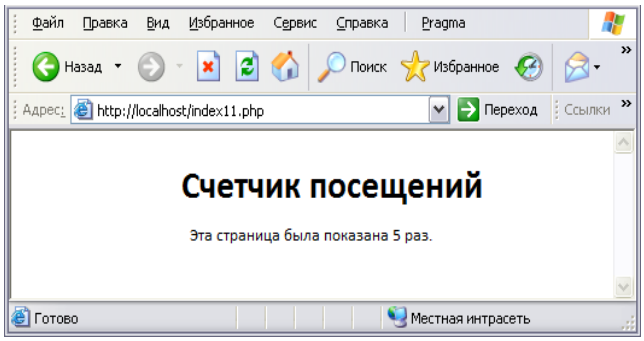


Рисунок 9.1 – Листинг программы счетчика (сессии) и результат его работы

Для создания счетчика посещений вы можете воспользоваться и другими возможностями, включая сохранение количества посещений во внешнем файле.

Еще один из способов заключается в использовании аутентификации пользователей и хранении данных об их посещении страниц в базах данных.

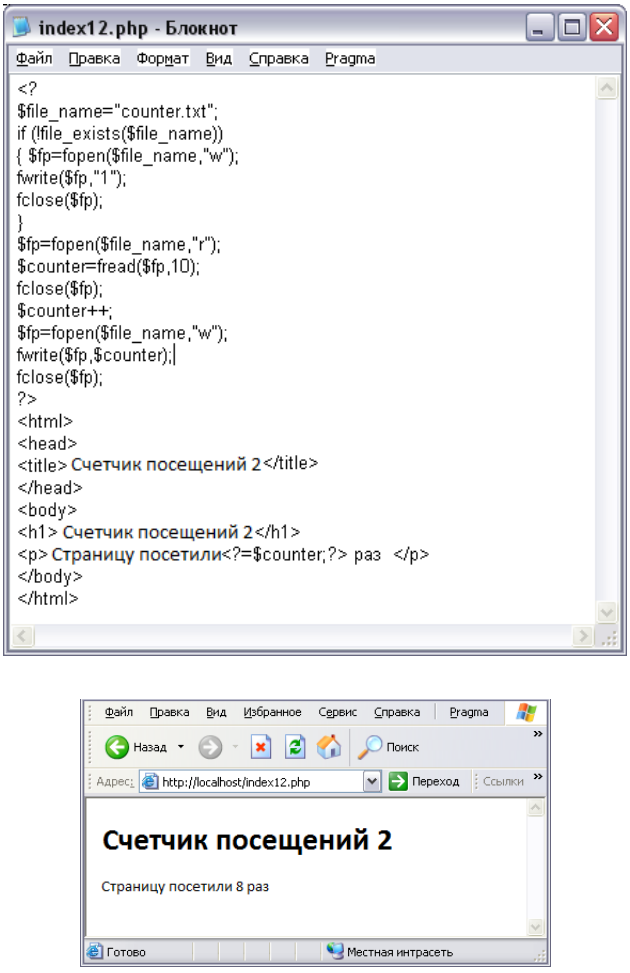


Рисунок 9.2 – Листинг программы счетчика (файл) и результат его работы

Контрольные вопросы

1. Что такое сеанс (сессия)?
2. Где хранится информация о сеансах?
3. Как открыть, сохранить и закрыть сеанс?
4. Как можно организовать счетчики посещений страницы?
5. Чем отличаются разные варианты счетчиков?

Лабораторная работа №10

Тема:PHP. Работа со сроками и регулярными выражениями

Цель:Научиться создавать на языкеPHP сценарии обработки срок и использование регулярных выражений,

Задание для выполнения

1. Ознакомиться с теоретическими сведениями об основных функциях обработки строк и работе с регулярными выражениями.

2. Создать строку S из Вашей фамилии, имени и отчества. Для этой строки
– определить коды символов, входящих в строку;
– удалить пробельные символы;
– найти позицию первого вхождения символов «а»(«о») и «с»(«к») в строку и их количество;
– «обрезать» строчку, начиная с Вашего имени;
– сравните строку со строкой, содержащей Вашу фамилию;
– найдите строку, содержащую 5 символов Вашей строки S, начиная с третьей;
– повторит строку, содержащую Ваше имя 3 раза;
– разделит строку S на массив из трех элементов;
– замените в строке S отчество на «Родитель»;
– «переверните» строку задом вперед;
– произведите перевод строки в верхний и нижний регистры, сделайте первые буквы каждого слова заглавными;
– найдите длину строки S и частоту составляемых символов.

3. Создайте веб-страницу, содержащую таблицу с информацией о студентах со следующими полями:
– фамилия и имя;
– дата рождения;
– адрес (город, улица);
- телефонный номер;
- группа.
Заполните ее 5-6 случайными записями.
С помощью регулярных выражений выведите список городов, где поживают студенты; даты рождения тех студентов, которым исполнилось более 21 года; номера телефонов МТС; удалите все тэги; *а также разделит первое поле на два – отдельно фамилию, отдельно имя.

Отчет должен содержать:

1. Тему, цель работы
2. Коды созданных файлов и копию окна обозревателя при их просмотре.

Методические указания

Строочные функции: Функции для работы с одиночными символами

chrВозвращает один символ с кодом.

Синтаксис:string chr(int ascii)

Возвращает строку из одного символа с кодом $code. Эта функция полезна для вставки каких-либо непечатаемых символов в строку – например кода нуля или символа прогона страницы, а также при работе с бинарными файлами.

Листинг 1. Пример использования функции chr

```
<?
// Сначала создаем массив того, что мы собираемся выводить,
// не заботясь о форматировании (дизайне) информации
    for($i=0, $x=0; $x<16; $x++) {
    for($y=0; $y<16; $y++) {
    $Chars[$x][$y]=array($i,chr($i));
    $i++; } }
// Теперь выводим накопленную информацию, используя идеологию
// вставки участков кода в HTML-документ
?>
<table border=1 cellpadding=1 cellspacing=0>
<?for($y=0; $y<16; $y++) {?> <tr>
<?for($x=0; $x<16; $x++) {?> <td>
<?=$Chars[$x][$y][0]?>:
<b><tt><?=$Chars[$x][$y][1]?></tt></b>
</td> <?}?> </tr> <?}?> </table>
```

0:	16: □	32:	48: 0	64: @	80: P	96: `	112: p	128: Ъ	144: ђ	160:	176: °	192: А	208: Р	224: а	240: р
1: □	17: □	33: !	49: 1	65: A	81: Q	97: a	113: q	129: Ѓ	145: `	161: Ў	177: ±	193: Б	209: С	225: б	241: с
2: □	18: □	34: "	50: 2	66: B	82: R	98: b	114: r	130: ‚	146: ’	162: ў	178: І	194: В	210: Т	226: в	242: т
3: □	19: □	35: #	51: 3	67: C	83: S	99: c	115: s	131: ѓ	147: “	163: Ј	179: і	195: Г	211: У	227: г	243: у
4: □	20: □	36: $	52: 4	68: D	84: T	100: d	116: t	132: „	148: ”	164: ¤	180: ґ	196: Д	212: Ф	228: д	244: ф
5: □	21: □	37: %	53: 5	69: E	85: U	101: e	117: u	133: …	149: •	165: Ґ	181: µ	197: Е	213: Х	229: е	245: х
6: □	22: □	38: &	54: 6	70: F	86: V	102: f	118: v	134: †	150: –	166: ¦	182: ¶	198: Ж	214: Ц	230: ж	246: ц
7: □	23: □	39: '	55: 7	71: G	87: W	103: g	119: w	135: ‡	151: —	167: §	183: ·	199: З	215: Ч	231: з	247: ч
8: □	24: □	40: (	56: 8	72: H	88: X	104: h	120: x	136: €	152: □	168: Ё	184: ё	200: И	216: Ш	232: и	248: ш
9:	25: □	41:)	57: 9	73: I	89: Y	105: i	121: y	137: ‰	153: ™	169: ©	185: №	201: Й	217: Щ	233: й	249: щ
10:	26: □	42: *	58: :	74: J	90: Z	106: j	122: z	138: Љ	154: љ	170: Є	186: є	202: К	218: Ъ	234: к	250: ъ
11:	27: □	43: +	59: ;	75: K	91: [	107: k	123: {	139: ‹	155: ›	171: «	187: »	203: Л	219: Ы	235: л	251: ы
12:	28: □	44: ,	60: <	76: L	92: \	108: l	124: |	140: Њ	156: њ	172: ¬	188: ј	204: М	220: Ь	236: м	252: ь
13:	29: □	45: -	61: =	77: M	93:]	109: m	125: }	141: Ќ	157: ќ	173:	189: Ѕ	205: Н	221: Э	237: н	253: э
14: □	30: □	46: .	62: >	78: N	94: ^	110: n	126: ~	142: Ћ	158: ћ	174: ®	190: ѕ	206: О	222: Ю	238: о	254: ю
15: □	31: □	47: /	63: ?	79: O	95: _	111: o	127: □	143: Џ	159: џ	175: Ї	191: ї	207: П	223: Я	239: п	255: я

Рисунок 10.1 – Результат работы программы листинга 1

ordВозвращает символ ascii код.

Синтаксис:int ord(string str) Эта функция возвращает ASCII код первого символа строки str. Например, ord(chr($n)) всегда равно $n – конечно, если $n помещено между нулем и 255.

Строчные функции : Функции отрезки пробелов

trimУдаляет из заданной строки начальные и конечные символы.

Синтаксис:string trim(string str)

Возвращает копию str только с удаленными ведущими и конечными пробольными символами. Под пробольными символами следует понимать "\n", "\r", "\t", "\v", "\0" и пропуск. Например, вызов trim("test\n") вернет строку "test".

ltrimУдаляет из заданной строки исходные символы.

Синтаксис:string ltrim(string str)

То же, что и trim(), только удаляет исключительно начальные пробельные символы ("\n", "\r", "\t", "\v", "\0" и пропуск), а конечные не трогает.

rtrimУдаляет из заданной строки конечные символы.

Синтаксис:string rtrim(string str)

То же, что и trim(), только удаляет исключительно конечные пробельные символы ("\n", "\r", "\t", "\v", "\0" и пропуск), а начальные не трогает. Эта функция – синоним chop().

chopУдаляет из заданной строки конечные символы.

Синтаксис:string chop(string str)

Удаляет только конечные пробелы, начальные не трогает.

Строчные функции: Поиск в тексте

strchrПоиск первого вхождения символа в строку.

Синтаксис:string strchr(string stroka, string str)

Данная функция работает идентично функции strstr().

strstrПоиск первого вхождения подстроки в строку.

Синтаксис:string strstr(string stroka, string str)

Функция strstr() возвращает фрагмент строки, заданной в параметре stroka, начиная с первого фрагмента, указанного в параметре str и до конца. В случае неудачи возвращает false. Данная функция чувствительна к регистру. Если str не является строкой, то значение преобразуется в целое и используется как код искомого символа.

Например, код

```
$email = " olena_my@mail.ru ";
$domain = strstr($email, "@");
echo $domain;
```

выведет @mail.ru

stristrНахождение первого вхождения подстроки, не считая регистра.

Синтаксис:string stristr(string stroka, string str)

Функция stristr() возвращает часть строки, заданной в параметре stroka, начиная с первого фрагмента, указанного в параметре str и до конца. В случае неудачи возвращает false. Данная функция нечувствительна к регистру.

Если str не является строкой, то значение преобразуется в целое и используется как код искомого символа.

strrchrПоиск последнего вхождения подстроки.

Синтаксис:string strrchr(string stroka, string str)

Функция strrchr() возвращает часть строки, заданную в параметре stroka, начиная с последнего фрагмента, указанного в параметре str и до конца. В случае неудачи возвращает false. Данная функция чувствительна к регистру. Если str не является строкой, то значение преобразуется в целое и используется как код искомого символа.

strposНаходит позицию первого вхождения подстроки в заданной строке.

Синтаксис:int strpos(string where, string what [, int fromwhere])

Функция strpos() пытается найти в строке were подстроку what и в случае успеха возвращает позицию (индекс) этой подстроки в строке. Первый символ строки имеет индекс 0. Необязательный параметр fromwhere можно задавать, если поиск нужно вести не с начала строки, а с другой позиции. В этом случае следует эту позицию передать в fromwhere. Если подстроку не удалось найти, функция возвращает false. Если параметр what не строчка, в этом случае его значение преобразуется в целое и используется как код искомого символа.

if(strpos($text, "a")==false) echo "Не найдено!";

strrposНаходит в заданной строке последнюю позицию, в которой находится заданный фрагмент.

Синтаксис:int strrpos(string where, string what)

Данная функция ищет в строке where последнюю позицию, в которой встречался символ what (если what – строчка из нескольких символов, то оказывается только первый из них, остальные не играют никакой роли). Если искомый символ первый в строке или вообще нет, функция вернет 0. В случае, если искомый символ не найден, возвращает false.

substr_countНаходит количество вхождений фрагмента в строку.

Синтаксис:int substr_count(string where, string what)

Функция substr_count() возвращает число фрагментов what, присутствующих в строке where.

```
echo substr_count("www.spravkaweb.ru", ".");
// Выведет 2
```

Строчные функции : Функции сравнения

strcmpСравнивает строчки.

Синтаксис:int strcmp(string str1, string str2)

Эта функция сравнивает две строки посимвольно (точнее, побайтово) и возвращает: 0 – если строки полностью совпадают; -1 – если строка str1 лексикографически меньше str2; 1 – если, напротив, str1 "больше" str2. Поскольку сравнение идет побайтово, то регистр символов влияет на результаты сравнений.

strncmpСравнивает начала строк.

Синтаксис:int strncmp(string str1, string str2, int len)

Эта функция отличается от strcmp() тем, что сравнивает не все слово целиком, а первые len байты. В случае если len меньше длины наименьшей из строк, то строки сравниваются целиком.

Поскольку сравнение идет побайтово, то регистр символов влияет на результаты сравнений.

strcasecmpСравнивает строчки без учета регистра.

Синтаксис:int strcasecmp(string str1, string str2)

То же, что и strcmp(), только при работе не учитывается регистр букв.

Строчные функции : Форматирование и вывод строк

printВыводит строку, значение переменной или выражение.

Синтаксис:print(string arg)

Функция print() выводит аргумент arg, какое может быть переменное или выражение.

echoПроводит вывод одного или нескольких значений.

Синтаксис:echo(string arg1, string [argn]...)

Функция echo() выводит значения перечисленных параметров. echo() – фактически языковая конструкция, поэтому для нее не обязательны скобки, даже если используется несколько аргументов.

printfВывод форматированной строки.

Синтаксис:int printf(string format [, mixed args...]);

Делает то же самое, что и sprintf(), только результирующая строка не возвращается, а направляется в браузер пользователя.

sprintfПроводит форматирование строки с подстановкой переменных.

Синтаксис:sprintf($format [,args...])

Эта функция возвращает строку, составленную на основе строки форматирования, содержащую некоторые специальные символы, которые впоследствии будут заменены на значение соответствующих переменных из списка аргументов.

Строка форматирования $format может включать команды форматирования, перед которыми стоит символ %. Остальные символы копируются в исходную строку как есть. Каждый спецификатор формата (т.е. символ % и последующие за ним команды) соответствуют одному и только одному параметру, указанному после параметра $format. Если же нужно поместить в текст % в качестве обычного символа, необходимо его удвоить:

echo sprintf("Состояние было %d%%",$%);

Каждый спецификатор формата включает в себя максимум пять элементов (в порядке их прохождения после символа %):

• Необязательный спецификатор размера поля, указывающий, сколько символов будет отведено под выводимую величину. Как символ-заполнитель (если значение имеет меньший размер, чем размер поля для его вывода) может использоваться пробел или 0, по умолчанию подставляется пробел. Можно указать другой символ, если указать его в строке форматирования.

• Опциональный спецификатор выравнивания, определяющий, будет результат выровнен по правому или по левому краю поля. По умолчанию производится выравнивание по правому краю, однако можно указать и левое выравнивание, задав символ – (минус).

• Необязательное число определяющее размер поля для вывода величины. Если результат в поле не будет помещаться, то он выйдет за края этого поля, но не будет усечен.

• Необязательное число, предшествующее точке ".", определяющее, сколько знаков после запятой будет в результирующей строке. Этот спецификатор учитывается только в том случае, если происходит вывод числа с плавающей точкой, иначе он игнорируется.

• Наконец, обязателен спецификатор типа величины, которая будет помещена в исходную строку:

• b – очередной аргумент из списка выводится как двоичное число

• c – выводится символ с указанным в аргументе кодом

• d – целое число
• f – число с плавающей точкой
• o – восьмеричное целое число
• s – строка символов
• x – шестнадцатеричное целое число с маленькими буквами az
• X – шестнадцатеричное целое число с прописными буквами AZ

Вот как можно указать точность представления чисел с плавающей точкой:

```
$money1 = 68.75;
$money2 = 54.35;
$money = $money1 + $money2;
// echo $money выведет "123.1"...
$formatted = sprintf ("%01.2f" $money);
// echo $formatted выведет "123.10"!
```

sscanfПроводит интерпретацию строки согласно формату и внесение значений в переменные.

Синтаксис:mixed sscanf(string str, string format [, string var1...])

Функция sscanf() является противоположностью функции printf(). Она интерпретирует строку str согласно формату format, аналогично спецификации printf().

Строочные функции: Сборка/разбиение строк

substrВозвращает участок строчки с определенной длиной.

Синтаксис:string substr(string str, int start [,int length])

Возвращает участок строки str, начиная с позиции start и длиной length. Если length не задан, то понимается подстрока от start к концу строки str. Если start больше, чем длина строки, или значение length равно нулю, то возвращается пустая подстрока. Однако эта функция может делать еще довольно полезные вещи. Например, если мы передадим в start отрицательное число, то будет считаться, что это число является индексом подстроки, но только отсчитываемым от конца str (например, -1 означает "начинается с последнего символа строки"). Параметр length, если он задан, тоже может быть отрицательным. В этом случае последним символом возвращаемой подстроки будет символ из str с индексом length, определяемым от конца строки.

```
$str = substr("abcdef", 1); // вернет "bcdef"
$str = substr("abcdef", 1, 3); // вернет "bcd"
$str = substr("abcdef"-1); // вернет "f"
$str = substr("abcdef"-2); // вернет "ef"
$str = substr("abcdef" -3, 1); // вернет "d"
$str = substr("abcdef", 1-1); // вернет "bcde"
```

str_repeatПовторяет строку определенного количества раз.

Синтаксис:string str_repeat(string str, int number)

Функция "повторяет" строку str number раз и возвращает объединенный результат.

```
echo str_repeat("test!",3); // выводит test!test!test!
```

str_padДополняет строчку другой строкой до определенной длины.

Синтаксис:string str_pad(string input, int pad_length [, string pad_string [, int pad_type]])

Аргумент input задает начальную строчку. Аргумент pad_length задает длину возвращаемой строки. Если он имеет значение меньше начальной строки, то никакого добавления не производится.

С помощью необязательного аргумента pad_string можно указать, какую строку использовать в качестве заполнителя (по умолчанию – пропуска). С помощью необязательного аргумента pad_type можно указать, с какой стороны следует дополнять строку: вправо, влево или с обеих сторон. Этот аргумент может принимать следующие значения:

• STR_PAD_RIGHT (по умолчанию)
• STR_PAD_LEFT
• STR_PAD_BOTH

```
$str = "Aaaaa";
echo str_pad($str, 10);
// вернет "Aaaaa"
echo str_pad($str, 10, "-=", STR_PAD_LEFT);
// вернет "-=-=-Aaaaa"
echo str_pad($str, 10, "_", STR_PAD_BOTH)
// вернет "_Aaaa_"
```

strtokВозвращает строчку по частям.

Синтаксис:string strtok(string arg1, string arg2)

Функция возвращает часть строки arg1 к разделителю arg2. При последующих вызовах возвращается следующая часть к следующему разделителю, и так к концу строки. При первом вызове функция принимает два аргумента: начальную строчку arg1 и разделитель arg2. При каждом последующем вызове аргумент arg1 указывать не следует, иначе будет возвращаться первая часть строки. Если возвращать больше нечего, функция вернет false. Если часть строки состоит из 0 или пустой строки, то функция также вернет false.

```
        $str="This is example№string№ Aaa";
        $tok = strtok($str, "");
        while($tok){ echo "$tok"; $tok = strtok("№");
};
// выведет: "This" "is" "an" "example" "string"
```

explodeПроизводит разделение строки в массив.

Синтаксис:array explode(string seperator, string str [, int limit])

Функция explode() возвращает массив строк, каждая из которых соответствует фрагменту начальной строки str, находящемуся между разделителями, указанными аргументом separator. Необязательный параметр limit указывает максимальное количество элементов в массиве. Оставшаяся неразделенная часть будет содержаться в последнем элементе.

```
$str = "Path1 Path2 Path3 Path4";
$str_exp = explode(" ", $str);
// теперь $str_exp = array([0] => Path1 [1] => Path2
// [2] => Path3 [3] => " [4] => Path4)
```

implodeПроизводит объединение массива в строчку.

Синтаксис:string implode(string glue, array pieces)

Функция implode() возвращает строку, содержащую последовательно все элементы массива, заданного в параметре pieces, между которыми вставляется значение, указанное в параметре glue.

$str = implode(":", $arr);

joinПроизводит объединение массива в строчку.

Синтаксис:string join(string glue, array pieces) То же, что и implode().

Строчные функции: Работа с блоками текста

str_replaceЗаменяет в начальной строке одни подстроки на другие.

Синтаксис:string str_replace(string from, string to, string str)

Эта функция заменяет в строке str все вхождение подстроки from (с учетом регистра) на to и возвращает результат. Начальная строка, передаваемая третьим параметром, при этом не меняется. также эта функция может работать с двоичными строчками.

substr_replaceЗаменяет в начальной строке одни подстроки на другие.

Синтаксис:string substr_replace(string str, string replacement, int start [, int lemgth])

Эта функция возвращает строку str, в которой часть символа с позицией start и длиной length (или до конца, если аргумент длины не указан) заменяется строкой replacement. Если значение start положительно, то отсчет производится от начала строки str, иначе – от конца (-1 – последний символ строки). Если значение length отрицательное, то оно указывает длину заменяемого фрагмента. Если оно отрицательное, то это число символов от конца строки str до последнего символа заменяемого фрагмента (со знаком минус).

wordwrapРазбивает исходный текст на строки с определенными символами.

Синтаксис:string wordwrap(string str [, int width [, string break [, int cut]]])

Эта функция разбивает блок текста str на несколько строк, завершающихся символами break, так, чтобы на одной строке было не более width буквы. Разбиение происходит по границе слова, так что текст остается читаемым.

strtrКомплексная замена в строке.

Синтаксис:string strtr(string str, string from, string to) string strtr(string str, array from)

В первом случае функция strtr() возвращает строку str, в которой каждый символ, присутствующий в строке from, заменяется на соответствующую строке to. В случае если строки from и to разной длины, то лишние конечные символы длинной строки игнорируются. Во втором случае функция strtr() возвращает строку, в которой фрагменты строки str заменяются на соответствующие индексам значения элементов массива from. При этом функция пытается заменить первоначально самые большие фрагменты начальной строки и не производит замену в уже модифицированных частях строки. Таким образом, теперь мы можем выполнить несколько замен сразу.

strrevПроводит реверс строки.

Синтаксис:string strrev(string str)

Функция strrev() возвращает строку str "задом вперед".

Строковые функции: Функции для преобразования символов

nl2brЗаменяет символы перевода строки.

Синтаксис:string nl2br(string string)

Заменяет в строке все символы новой строки \n на
\n и возвращает результат. Начальная строка не меняется. Обратите внимание, что символы \r, присутствующие в конце строки текстовых файлов Windows, этой функцией никак не учитываются, а потому остаются на старом месте.

strip_tagsУдаляет из строки теги.

Синтаксис:string strip_tags(string str[, string allowable_tags])

Эта функция удаляет из строки все HTML- и PHP-теги и возвращает результат. Незавершенные или фиктивные тэги вызывают ошибку. В параметре allowable_tags можно передать теги, которые не следует удалить из строки. Они должны перечисляться вплотную друг к другу.

get_meta_tagsФункция ищет и обрабатывает все теги <META>.

Синтаксис:array get_meta_tags(string filename, int use_include_path) Функция открывает файл и ищет в нем все теги <META> до тех пор, пока не встретится закрывающий тег</head>.

Строочные функции: Функции изменения регистра

strtolowerПроводит преобразование символов строки в нижний регистр.

Синтаксис:string strtolower(string str);

Преобразует строку в нижний регистр. Возвращает результат перевода.

```
$str = "HeLLo World";
$str = strtolower($str);
echo $str;
// выведет hello world
```

strtoupperПроводит преобразование заданной строки в верхний регистр.

Синтаксис:string strtoupper(string str); Перевод строки в верхний регистр. Возвращает результат преобразования.

```
$str = "Hello World";
$str = strtoupper($str);
echo $str;
// выведет HELLO WORLD
```

ucfirstПроводит преобразование первого символа строки в верхний регистр.

Синтаксис:string ucfirst(string str);

Возвращает строку, у которой первый символ заглавный.

```
$str = "hello world";
$str = ucfirst($str);
echo $str;
// выведет Hello world
```

ucwordsПроводит преобразование первого символа каждого слова строки в верхний регистр.

Синтаксис:string ucwords(string str);

Возвращает строку, у которой первый символ каждого слова в строке заглавный.

```
$str = "hello world";
$str = ucfirst($str);
```

```
echo $str;
// выведет Hello World
```

Строчные функции : Строчные суммы

strlenВозвращает длину строки.

Синтаксис:int strlen(string str)

Возвращает просто длину строки, то есть сколько символов содержится в str. Строка может содержать любые символы, в том числе с нулевым кодом. Функция strlen() будет правильно работать и с такими строками.

count_charsВозвращает информацию о символах строки.

Синтаксис:mixed count_chars(string str[, int mode])

Функция count_chars() подсчитывает частоту каждого байта (0-255) в строке str и возвращает в массиве результат согласно необязательному аргументу mode. mode может принимать следующие значения:

• 0 (по умолчанию) – массив с байтами в качестве индексов и частотой повторения как значение элемента массива

• 1 – похож на 0, но отсутствуют в строке str байты не возвращаются

• 2 – похож на 0, но возвращаются только те байты, которые отсутствуют

• 3 – возвращающаяся строка, состоящая из всех обнаруженных символов

• 4 – возвращается строка, состоящая из всех отсутствующих символов

Понятие регулярного выражения

Регулярное выражение (regular expression, сокращенно)РО) – это технология, позволяющая задать шаблон и осуществить поиск данных, соответствующих этому шаблону, в заданном тексте, представленном в виде строки.

Кроме того, с помощью регулярных выражений можно изменить и удалить данные, разбить строку по шаблону на подстроки и многое другое.

Одно из распространенных применений РО– это проверка строки на соответствие каким-либо правилам. Например, следующее РО предназначен для проверки того, что строка содержит корректный e-mail-адрес:

```
/^\w+([\.\w]+)*\ w@ \w((\.\w)*\w+)*\.\w{2,3}$/
```

Давайте подумаем, что является корректным e-mail-адресом. Это набор букв, цифр и символов подчеркивания, после которых следует специальный символ «собака» @, затем еще один такой же набор, содержащий имя сервера, точку (.) и две или три буквы, указывающие на зону домена, к которой принадлежит почтовый ящик (ru, com, org и т.д.). Приведенный выше РО формализует данное описание на языке, понятном компьютеру. И описывает не какой-то конкретный электронный адрес, а все возможные корректные электронные адреса. Таким образом, производится формальная задача множества правильных e-mail'ов с помощью шаблона регулярного выражения. Другие примеры шаблонов – это шаблоны MS Word и HTML-формы.

Механизм регулярных выражений задает правила построения шаблонов и производит поиск данных по этому шаблону в указанной строке.

В дальнейшем изложении термины РО и «шаблон» часто будут использоваться в качестве синонимов, но важно понимать, что это не совсем одно и то же. Шаблон за-

дает какой-то тип данных, аРО– это механизм, который производит поиск и включает шаблон и опции поиска, а также задает язык написания шаблонов.

Регулярные выражения в PHP

Регулярные выражения пришли из UNIX и Perl. Как упоминалось выше, с помощью регулярных выражений можно искать и изменять текст, разбивать строчку на подстроки и т.д. В PHP существуют такие удобные и мощные средства работы со строками, как explode (разбиение строки на подстроки), strstr (нахождение подстроки), str_replace (замена всех вхождений подстроки). Возникает вопрос – зачем придумывать что-нибудь еще?

Основное преимуществоРОзаключается в том, что они позволяют организовать более гибкий поиск, то есть найти то, о чем нет точного знания, но есть образцовое представление. К примеру, нужно найти все семизначные номера телефонов, встречающихся в тексте. Мы не ищем какой-нибудь заранее известный нам номер телефона, мы знаем только, что искомый номер состоит из семи цифр. Для этого можно воспользоваться следующимРО:

/\d{3}-\d{2}-\d{2}/m

В PHP существует два разных механизма для обработки регулярных выражений: POSIX-совместимые и Perl-совместимые (сокращенно PCRE). Их синтаксис во многом схож, однако Perl-совместимые регулярные выражения более мощные и к тому же работают гораздо быстрее. Начиная с версии PHP 4.2.0, PCRE входят в набор базовых модулей и подключены по умолчанию. POSIX-совместимыеРОвключены по умолчанию только в версию PHP для Windows.

Основные функции для работы с Perl-совместимыми регулярными выражениями:

preg_match(pattern, string [result, flags]) и

preg_match_all(pattern, string, result [flags])>,

где:

pattern – шаблон регулярного выражения;

string – строка, в которой производится поиск;

result – содержит массив результатов (нулевой элемент массива содержит соответствие всему шаблону, первый – первому «восторженному» подшарлону и т.д.);

flags – необязательный параметр, определяющий, какие упорядоченные результаты поиска.

Эти функции осуществляют поиск по шаблону и возвращают информацию о том, сколько раз произошло совпадение. Для preg_match() это 0 (нет совпадений) или 1, поскольку поиск прекращается, как только найдено первое совпадение. Функция preg_match_all() производит поиск до конца строки и поэтому находит все совпадения. Все точные совпадения содержатся в первом элементе массива result в каждой из этих функций (для preg_match_all() этот элемент тоже массив).

Аналогом preg_match является булевая функция POSIX-расширения

ereg(string pattern, string string [, array regs])

Функция ereg() возвращает TRUE, если совпадение найдено, и FALSE – иначе.

Следующие примеры можно тестировать на перечисленных функциях. Например, так:

Листинг 2 Пример работы регулярного выражения

```
<?
// строка, в которой нужно что-то найти
$str = "Мой телефонный номер: 33-22-44".
"Номер моего редактора: 222-44-55 и 323-22-33";
// шаблон, по которому искать.
// Задает поиск семизначных номеров.
$pattern = "/\d{3}-\d{2}-\d{2}/m";
// функция, осуществляющая поиск
$num_match = preg_match_all ($pattern, $str, $result);
//вывод результатов поиска
for ($i=0; $i < $num_match; $i++)
echo "Совпадение $i: ".$result[0][$i]."<br>";
?>
```

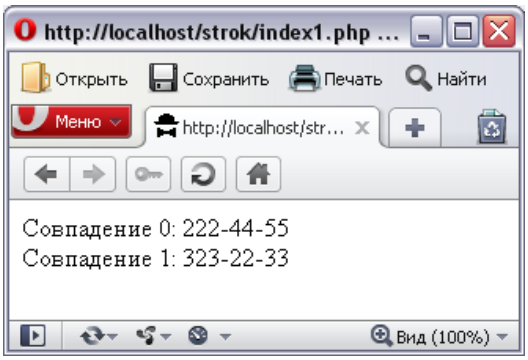


Рисунок 10.2 – результат работы программы листинга 2

Синтаксис регулярных выражений

Регулярное выражение является строчкой. Эта строка состоит из собственно регулярного выражения (шаблона), выделенного с помощью специального символа разделителя (это могут быть символы "/", "|", "{", "!" и т.д.) и модификатора, влияющего на способ обработки РО.

Например, в регулярном выражении /\d{3}-\d{2}-\d{2}/m символ "/" является разделителем, \d{3}-\d{2}-\d{2} – непосредственно регулярное выражение (шаблон), а m – модификатор.

Мощность регулярных выражений порождена в основе своей способностью включать в шаблон альтернативы и повторения. Они кодируются в шаблоне с помощью метасимволов. Метасимвол отличается от любого другого символа тем, что имеет особое значение.

Одним из основных метасимволов является обратный слеш «\». Он меняет тип символа, следующего за ним, на противоположный, то есть если это был обычный символ, то он может превратиться в метасимвол, если это был метасимвол, то он теряет свое специальное значение и становится обычным символом (это нужно для того, чтобы вставлять в текст (специальные символы как обычные). Например, символ d в обычном режиме не имеет никаких специальных значений, но есть метасимвол, означающий «любая цифра». Символ «.» в обычном режиме означает "любой единичный символ", а "\." означает просто точку.

Другое назначение обратного слеша – кодирование специальных символов, таких как:

\n – символ перевода строки;

\e – символ escape;

\t – символ табуляции;

\xhh – символ в шестнадцатеричном коде, например, \x41 есть буква A и т.д.

Еще одно предназначение обратного слеша – обозначение генерируемых символьных типов, таких как:

\d – любая десятичная цифра (0-9);

\D – любой символ, не являющийся десятичной цифрой;

\s – любой пустой символ (пробел или табуляция);

\S – любой символ, не пустой;

\w – символ, используемый для написания Perl-слов (это буквы, цифры и символ подчеркивания), так называемый «словарный символ»;

\W – несловесный символ (все символы, кроме определяемых \w).

Что подразумевается под «символьным типом»? Просто каждый метасимвол принимает значение (одно) из класса возможных значений, заданных автоматически или вручную. Символьные типы, задаваемые пользователем, описываются с помощью квадратных скобок.

Пример использования приведенных выше метасимволов:

/\d\d\d plus \d is \w\w\w/

Этот РО означает: трехзначное число, за которым следует подстрока plus, любая цифра, затем is и слово из трех словарных символов. В частности, данным РО удовлетворяют строки: "123 plus 3 is sum", "213 plus 4 is 217".

Вообще различают два множества метасимволов: распознающиеся в любом месте шаблона, за исключением внутренности квадратных скобок, и распознающиеся внутри квадратных скобок.

Квадратные скобки [ ] применяются для описания подмножеств и внутри регулярного выражения рассматриваются как один символ, который может принимать значения, перечисленные внутри этих скобок. Однако если первым символом внутри скобок является ^, то значением символьного класса могут быть только символы, не перечисленные внутри скобок.

Примеры:

Символьный класс [абвгд] задает один из символов а, б, в, г, д, а класс [^абвгд] задает любой символ, кроме а, б, в, г, д.

Если написать [2бул]ки], то это выражение интерпретируется как один из символов 2, б, у, л, за которым следует строка ки], так как первая закрывающая квадратная скобка (разбор происходит слева направо), которая заканчивается класса. То есть это РО совпадет с одной из строк 2ки], бки], уки] или лки].

С помощью РО [0-9А-Яа-я] можно задать любую букву или цифру.

Метасимволы, распознаваемые вне квадратных скобок, можно разделить на группы следующим образом: определяющие положение искомого текста в строке, связанные с подвыражениями, ограничивающие символьный класс, квантификаторы и перечисление альтернатив.

Таблица 10.1 – Метасимволы, распознающиеся изнутри квадратных скобок

Метасим-	Значение

вол	
\	Переходный символ с множеством назначений
^	Возражение класса, но если это первый символ (например, «^\d» задает все, кроме цифр)
-	Задает диапазон символов (например, 0-9 задает все цифры, AZ – все латинские буквы)
]	Вычисляет сивольный класс

Регулярное выражение /d/d может быть сопоставлено следующим подстрокам: 11, 22, 33. Если в начале РО стоит ^, то совпадения ищутся в начале строки, поэтому выражение /^d/d найдет только 11.

Когда в конце РО стоит знак доллара $, поиск производится в конце строки, поэтому выражение /d$d найдет только 33.

Шаблона же /^\d\d\d$/ будет удовлетворять строку, которая целиком состоит из трехзначного числа (т.е. она и начинается и заканчивается этим числом).

Найдем все html-теги, расположенные в начале каждой строки файла 1.htm.

```
<?php
//считываем файл в сроке
$str = file_get_contents('1.htm');
$pattern = "!^<[^/]+>!mU";
// осуществляем поиск
$n = preg_match_all($pattern, $str, $res);
// выводим результаты
for($i = 0; $i < $n; $i++) echo htmlspecialchars($res[0][$i])."<br>";
?>
```

Шаблон ограничен восклицательными знаками. Первая «^» означает, что мы ищем совпадения в начале строк, затем следует символ «<» – его и ищем в строке, после него должно идти все, что угодно, кроме слеша (конструкция «[^/]»), «+ » говорит, что стоящий перед ним символ повторяется один и более раз и заканчивается все это символом «>». Таким образом, выделяются все тэги в начале строк.

Таблица 10.2 – Метасимволы, распознающиеся вне квадратными скобками

Мета-символ	Значение
\	Переходный символ с множеством назначений
^	Объявляет начало объекта (или строчки в многострочном режиме). То есть, этот символ определяет, что искомый текст должен находиться в начале строки. Альтернатива: "\A"

$	Объявляет конец объекта (или строчки в многострочном режиме). То есть, этот символ определяет, что искомый текст должен находиться в конце строки. Альтернативы: \Z, \z
.	Совпадает с любым символом, кроме символа перевода строки (по умолчанию)
[	Начинает определение символьного класса
]	Заканчивает определение символьного класса
|	Разделяет перечисление альтернативных вариантов
(	Начинает подшаблон регулярное (подвыражение)
)	Заканчивает подшаблон
?	Расширяет значение "(", квантификаторов 0 или 1, и квантификатор минимизации).
*	0 или более повторений (квантификатор)
+	1 или более повторений (квантификатор)
{	Начинает минимальный/максимальный квантификатор
}	Кончает минимальный/максимальный квантификатор

Мы хотим убедиться, что имя автора было записано правильно (сначала фамилия с прописной буквы, затем инициалы через точку) и находится в конце строки.

```
<?php
// считываем файл в сроке
$str = file_get_contents('1.htm');
$pattern = "!\s[А-Яа-я]+\s([А-Я]\.\s*)([А-Я]\.\s*)$!m";
// шаблон ограничен восклицательными знаками,
// m – модификатор, включающий многострочный режим
// первый означает, что перед фамилией должен идти пустой символ (например,
//пробел)
// [А-Яа-я] задает одну из букв алфавита в любом регистре,
// а в комбинации со знаком плюс определяет, что эта буква повторяется один и более раз.
// следующий \s означает, что между фамилией и инициалами должен быть пробел
// Далее следует подвыражение, определяющее инициалы.
// Это буква от А до Я, после которой стоит точка ('\.')
// Экранируем точку, чтобы избавиться от ее специального значения.
// После буквы с точкой может идти или не идти пробел или несколько.
// Вся конструкция повторяется минимум в два раза.
// Последний символ $ означает, что фамилия с инициалами
// должны находиться в конце срока.
```

```
// Осуществляем поиск
    $n = preg_match_all($pattern, $str, $res);
    // выводим результаты
    for ($i = 0; $i < $n; $i++) echo htmlspecialchars($res[0][$i])."<br>";
    ?>
```

Примеры (| и .)

Пусть есть какой-нибудь текст. Нам нужно найти всех упомянутых в нем людей с учеными званиями.

```
    <?
    $str = "Доцент Смирнов совершил открытие.".
    "Его учителем была профессор Иванова.".
    "Этим открытием Смирнов завоевал на себя степень доктора.".
    "Раньше он был только кандидат.";
    $pattern = "/(профессор|доцент)\s[А-Яа-я]+(\s|\.)/i";
    // осуществляем поиск
    $n = preg_match_all($pattern, $str, $res);
    // выводим результаты
    for($i = 0; $i < $n; $i++) echo htmlspecialchars($res[0][$i])."<br>";
    ?>
```

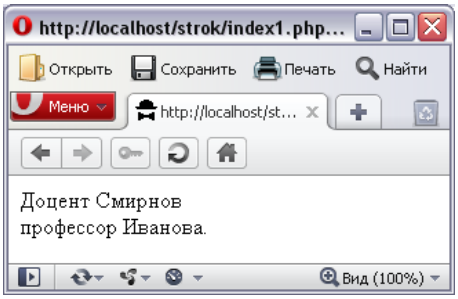


Рисунок 10.3 – Результат работы скрипта

Метасимвол прямой предел » позволяет задавать альтернативные варианты. В нашем примере мы хотели найти всех профессоров или доцентов. Для этого было создано подвыражение (профессор | доцент). После звания через пропуск фамилию человека, которому она принадлежит, для этого существует комбинация «s[А-яа-я]+». После фамилии идет либо снова пропуск, либо точка, если это конец предложения. Получаем снова два альтернативных варианта: «(\s|\.)» (здесь точка экранируется обратным слешем, чтобы она понималась как обычная точка, без особого значения).

Выражения (подшаблоны)

В РО подшаблоны выделяют, беря в круглые скобки. Для их обозначения кроме термина «подшаблон» используют термин «подвыражение». Подшаблоны могут быть вложены. Выделение части регулярного выражения посредством регулярного подвыражения выполняет следующее.

– Локализует множество альтернатив.

Например, шаблон

```
жар(кое|птица|)
```

совпадает с одним из слов «жаркое», «жарптица» и «жар». В то время как без скобок это было бы «жаркое», «птица» и пустая строчка.

– устанавливает подшаблон как «захватывающий» подшаблон. Это означает, что, когда какая-то подстрока в тексте совпала с шаблоном, все подстроки, совпавшие с подшаблонами этого РО, тоже возвращаются как результат. Скобки, обозначающие начало подшаблона, перечисляются слева направо (начиная с 1) для того, чтобы узнать сколько подшаблонов нужно захватить.

Например, есть такой шаблон:

победитель получит ((золотую|позолоченный)(медаль|кубок))

и строчка, в которой ищутся совпадения с этим шаблоном: «победитель получит золотую медаль». Тогда кроме этой фразы будут еще увлечены и изданы в результате поиска следующие совпадения в подвыражениях: «золотую медаль», «золотую», «медаль», пронумерованные 1, 2, 3 соответственно.

Однако это не всегда удобно. Для того чтобы избавиться от «захватывающего» эффекта подвыражения, после открывающей скобки пишут «?:». Тогда это подвыражение в результат поиска не включается и при нумерации остальных подшаблонов с «захватывающим» эффектом не учитывается.

победитель получит ((?:золотую|позолоченный)(медаль|кубок))

Тогда в условиях предыдущего примера получим разыскиваемую строку «победитель получит золотую медаль» и строки «золотую медаль», «медаль», пронумерованные 1 и 2 соответственно.

Повторение (квантификаторы)

В предыдущих примерах часто использовались комбинации типа \d\d. Это означает, что цифра должна повторяться дважды. А что делать, если повторений очень много или мы не знаем, сколько именно? Оказывается, нужно использовать специальные метасимволы.

Повторения описываются с помощью так называемых квантификаторов (метасимволов, задающих количественные отношения). Существует два типа квантификаторов: общие (задаются с помощью фигурных скобок) и сокращенные (это сокращение наиболее распространенных квантификаторов).

Квантификаторы могут следовать любому из перечисленных элементов:

– одиночный символ (возможно, в комбинации с обратным слешем);

– метасимвол «точка»;

– символьный класс;

– обратная ссылка;

– подшаблон.

Общие квантификации задают минимальное и максимальное число разрешенных повторений элемента; эти два числа, разделенные запятой, берутся в фигурные скобки. Числа не должны превышать 65 536 и первое число должно быть меньше или равно второму. Например

x{1,3}

говорит о том, что символ x должен повторяться минимум один, а максимум три раза. Соответственно этому шаблону удовлетворяют строки: x, xx, xxx.

Если второй параметр отсутствует, но запятая есть, то повторений может быть сколько угодно. Таким образом

[aeuoi]{2}

означает, что любой из символов "a", "e", "u", "o", "i" в строке может повторяться два и более раза, а регулярное выражение

\d{3}

задает ровно три цифры.

Сокращенные квантификации задают наиболее используемые количественные отношения (повторения). Они придуманы для удобства, чтобы не перегружать и без того сложные выражения излишним синтаксисом.

Исходя из исторических традиций три наиболее часто встречающихся квантификатора имеют следующие обозначения:

* эквивалентно {0,} – то есть это ноль и более повторений;

+ эквивалентно {1,} – то есть это одно и более повторений;

? эквивалентно {0,1} – то есть это ноль или одно повторение.

Есть еще один немаловажный момент, на который следует обратить внимание при изучении квантификаторов. По умолчанию все квантификаторы «алчны», они стремятся захватить как можно больше повторений элемента. То есть, если указать, что символ должен повторяться один и более раз (например, с помощью *), совпадение состоится со строкой, содержащей наибольшее число повторений указанного символа. Это может создать проблемы, например, при попытке выделить комментарии в программе на языке Ci или PHP. Комментарии в Ci и PHP записываются между символами /* и */, внутри которых тоже могут встречаться символы * и /. И попытка выявить Си-комментарии с помощью шаблона

/* .* */

в строке

/* первый комментарий */

не комментарий

/* второй комментарий */

не увенчается успехом из-за «жадности» элемента «.*» (будет найдена также строка «не комментарий»).

Для решения этой проблемы нужно написать вопросительный знак после квантификатора. Тогда он перестанет быть «алчным» и попытается захватить как можно меньшее число повторений элемента, к которому он применен (квантификатор применяется к стоящему перед ним элементу). Следовательно шаблон

/* .*? */

успешно выделяет Си-комментарии.

В PHP существует опция PCRE_UNGREEDY, которая делает все квантификаторы «не жадными» по умолчанию и «жадными», если после них следует вопросительный знак.

Листинг 3

```
<?
// Рассмотрим html-файл, где имеется следующая строка:
```

```
$str = "<div id=1>Привет</div><p>Текст, не заключенный в тег div</p><div id=2>Пока</div>";
// Если мы хотим найти текст,
// содержащийся между тегами div,
// естественно написать такой шаблон:
$pattern = "!<div id=1>.*</div>!si";
// Но этот шаблон слишком "жадный" и захватит также и текст,
// заключенный в нашем примере между тегами <p>.
// Чтобы этого избежать, нужно написать следующий шаблон,
// отличающийся только наличием знака вопроса,
// который запрещает квантификатору быть "жадным".
$pattern1 = "!<div id=1>.*?</div>!si";
// Запускаем поиск в строке $str совпадающий
// с шаблонами: $pattern и $pattern1
preg_match_all ($pattern, $str, $res);
preg_match_all ($pattern1, $str, $res1);
// выводим результаты поиска
// функция htmlspecialchars позволяет выводить html
// без его обработки браузером
echo "Нет шаблона:".htmlspecialchars($res[0][0])."<br>";
echo "Нежелательный шаблон:".htmlspecialchars($res1[0][0]);
?>
```

Результаты работы скрипта:

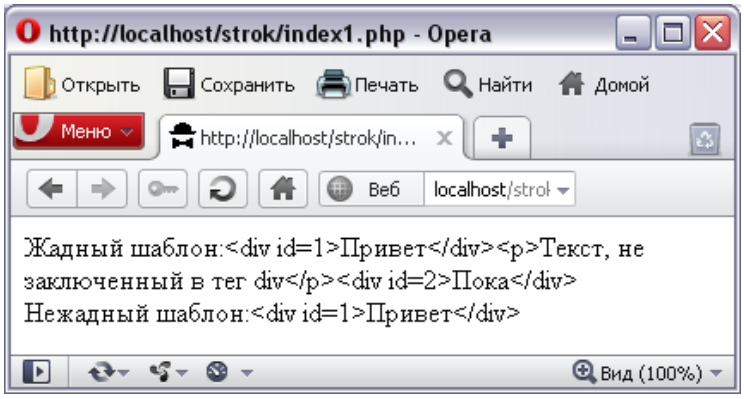


Рисунок 10.4 – Результат работы скрипта листинга 3

Модификаторы PCRE

Еще один важный элемент регулярного выражения – это список применяемых к нему модификаторов. Модификаторы – это выдаваемая интерпретатору регулярных выражений инструкция по обработке данного выражения. Например, считать, что все символы регулярного выражения соответствуют как прописным, так и маленьким буквам в строке, где производится поиск. Примеры модификаторов приведены в таблице 10.3.

Таблица 10.3 – Наиболее часто используемые модификаторы	
Обозначение	Описание
i (PCRE_CASELESS)	Если указан этот модификатор, то буквы в шаблоне совпадают с буквами и верхнего, и нижнего регистра в строке
m (PCRE_MULTILINE)	По умолчанию строка, подаваемая на вход интерпретатоуPO, рассматривается как составленный из одной линии. Этот модификатор включает поддержку многострочного режима
s (PCRE_DOTALL)	Если этот модификатор установлен, то метасимвол точка «.» совпадает с любым символом, ВКЛЮЧАЯ символ перевода строки
x (PCRE_EXTENDED)	Заставляет интерпретатор игнорировать пробелы между символами в шаблоне, за исключением пробелов, экранированных обратным слешем или находящихся внутри символьного класса, а также между неэкранированным символом # вне символьного класса и символом новой строки
U (PCRE_UNGREEDY)	Этот модификатор инвертирует «жадность» квантификаторов, то есть они становятся «нежадными» по умолчанию и «алчными» если предшествуют символу «?»

Контрольные вопросы

1. Как в PHP определяется строка символов?
2. Какие основные функции для создания, поиска, сравнения, преобразования, разделения, определения параметров строк Вам известны? Каков их синтаксис?
3. Что такое регулярные выражения? Зачем они используются?
4. Какие виды синтаксиса регулярных выражений существуют? В чем их отличие?
5. Что такое метасиволы? Какие цели символы существуют? Как они интерпретируются? Что это значит?
6. Что такое квантификатор? Зачем используется? Приведите примеры.
7. Что такое модификаторы? Зачем они предназначены? Приведите примеры.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Кузнецов М. В., Симдянов И. В. Самоучитель PHP 5.– СПб.: БХВ-Петербург, 2006.– 608 с.
2. Дамашке Г. PHP и MySQL. – М.: НТ Пресс, 2008. – 314 с.
3. Стеймец В., Вард Б. PHP: 75 готовых решений для вашего Web-сайта. – СПб.: Наука и техника, 2009. – 256 с.
4. Колисниченко, Денис PHP и MySQL. Разработка Web-приложений / Денис Колисниченко. - М.: БХВ-Петербург, 2013. - 560 с.
5. Колисниченко, Денис Профессиональное программирование на PHP (+CD-ROM) / Денис Колисниченко. - М.: БХВ-Петербург, 2015. - 416 с.
6. Кузнецов, М. PHP. Практика создания Web-сайтов / М. Кузнецов, И. Симдянов. - М.: БХВ-Петербург, 2012. - 577 с.
7. Кузнецов, М. Объектно-ориентированное программирование на PHP / М. Кузнецов, И.

Учебное издание

МЕТОДИЧЕСКИЕ УКАЗАНИЯ
К ВЫПОЛНЕНИЮ ЛАБОРАТОРНЫХ РАБОТ
по дисциплине
«КОМПЬЮТЕРНЫЕ И ТЕЛЕКОММУНИКАЦИОННЫЕ ТЕХНОЛОГИИ В ПРОФЕССИОНАЛЬНОЙ ДЕЯТЕЛЬНОСТИ»
для студентов направления подготовки
Профессиональное обучение (по отраслям),
магистерская программа
«Информационные технологии и системы» (в 2 х частях). Часть 2

С о с т а в и т е л ь:
Тимошенко Дарья Сергеевна

Печатается в авторской редакции.
Компьютерная верстка и оригинал-макет автора.

Подписано в печать _____
Формат 60х841/16. Бумага типограф. Гарнитура Times
Печать офсетная. Усл. печ. л. _____. Уч.-изд. л. _____
Тираж 100 экз. Изд. №_____. Заказ №_____. Цена договорная.

Издательство Луганского государственного
университета имени Владимира Даля

Свидетельство о государственной регистрации издательства
МИ-СРГ ИД 000003 от 20 ноября 2015г.

Адрес издательства: 91034, г. Луганск, кв. Молодежный, 20а
Телефон: 8 (0642) 41-34-12, факс: 8 (0642) 41-31-60
E-mail: izdat.lguv.dal@gmail.com **http:** //izdat.dahluniver.ru/